NAVAL POSTGRADUATE SCHOOL
Monterey, California

2

THESIS

ADAPTIVE DIGITAL NOTCH FILTERING

by

Kaluri Venkata Rangarao

September 1991

Thesis Advisor	M.A. Soderstrand
Co-Advisor	H. Loomis

Unclassified

security classification of this page

REPORT DOCUMENTATION PAGE

1a Report Security Classification Unclassified	1b Restrictive Markings
2a Security Classification Authority	3 Distribution/Availability of Report
2b Declassification/Downgrading Schedule	Approved for public release; distribution is unlimited.
4 Performing Organization Report Number(s)	5 Monitoring Organization Report Number(s)
6a Name of Performing Organization Naval Postgraduate School — 6b Office Symbol *(if applicable)* 3A	7a Name of Monitoring Organization Naval Postgraduate School
6c Address *(city, state, and ZIP code)* Monterey, CA 93943-5000	7b Address *(city, state, and ZIP code)* Monterey, CA 93943-5000
8a Name of Funding/Sponsoring Organization — 8b Office Symbol *(if applicable)*	9 Procurement Instrument Identification Number
8c Address *(city, state, and ZIP code)*	10 Source of Funding Numbers

Program Element No	Project No	Task No	Work Unit Accession No

11 Title *(include security classification)* ADAPTIVE DIGITAL NOTCH FILTERING

12 Personal Author(s) Kaluri Venkata Rangarao

13a Type of Report	13b Time Covered	14 Date of Report *(year, month, day)*	15 Page Count
Master's Thesis	From To	September 1991	103

16 Supplementary Notation The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government, or Indian Government.

17 Cosati Codes

Field	Group	Subgroup

18 Subject Terms *(continue on reverse if necessary and identify by block number)* estimation, spectrum estimation, identification ,notch filtering,adaptive filtering, signal processing

19 Abstract *(continue on reverse if necessary and identify by block number)*

The problem of narrow band interference while transmitting broad-band signals like Direct Sequence Spread Spectrum is a common source of problems in Electronic Warfare. This can occur either due to intentional jamming or due to unavoidable signal sources present in the vicinity of the receiver. Lack of improper information on these narrow band interferers makes it difficult to cancel them.

In this thesis the above problem is addressed by using an adaptive notch filtering technique. Before adopting such a technique other methods like the Least Square Estimator and the Maximum Likelihood Estimator were explored. However the Kwan and Martin adaptive notch filter structure was found both relevant and suitable for the problem of interest. The Kwan and Martin method has the difficulty of increasing hardware complexity with number of notches. This makes it difficult to implement in real time. A new algorithm was developed for the purpose of implementing the structure in real-time. This new algorithm offers the same performance at reduced hardware complexity. This algorithm was simulated and the results were presented. A hardware feasibility is discussed by proposing a simple structure based upon existing commercial signal processing chips.

20 Distribution/Availability of Abstract [✓ unclassified/unlimited] ☐ same as report ☐ DTIC users	21 Abstract Security Classification Unclassified	
22a Name of Responsible Individual H.Loomis	22b Telephone *(Include Area code)* (408) 646-[illegible] 3214	22c Office Symbol EC/Lm

DD FORM 1473,84 MAR   83 APR edition may be used until exhausted
All other editions are obsolete

security classification of this page

Unclassified

Adaptive Digital Notch Filtering

by

Kaluri V Rangarao
BSEE., ME(Automation) .,
Scientist 'E', DRDO India.

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN SYSTEMS ENGINEERING
(Electronic Warfare)

from the

NAVAL POSTGRADUATE SCHOOL

September 1991

Author: [illegible]

Kaluri V Rangarao

Approved by: Michael A. Soderstrand

Michael Soderstrand, Thesis Advisor

[illegible]

Herschel Loomis, Co-Advisor

[illegible]

Joseph Sternberg, Chairman
Electronic Warfare Academic Group

ABSTRACT

The problem of narrow band interference while transmitting broad-band signals like Direct Sequence Spread Spectrum is a common source of problems in Electronic Warfare. This can occur either due to intentional jamming or due to unavoidable signal sources present in the vicinity of the receiver. Lack of improper information on these narrow band interferers makes it difficult to cancel them.

In this thesis the above problem is addressed by using an adaptive notch filtering technique. Before adopting such a technique other methods like the Least Square Estimator and the Maximum Likelihood Estimator were explored. However the Kwan and Martin adaptive notch filter structure was found both relevant and suitable for the problem of interest. The Kwan and Martin method has the difficulty of increasing hardware complexity with number of notches. This makes it difficult to implement in real time. A new algorithm was developed for the purpose of implementing the structure in real-time. This new algorithm offers the same performance at reduced hardware complexity. This algorithm was simulated and the results were presented. A hardware feasibility is discussed by proposing a simple structure based upon existing commercial signal processing chips.

TABLE OF CONTENTS

I. INTRODUCTION 1

A. MOTIVATION 1

1. Electronic Warfare(EW) 1

2. Assumptions 1

B. PROBLEM DEFINITION 3

II. ESTIMATION TECHNIQUES 6

A. DIGITAL FILTERS 6

1. Some Hardware Schemes 6

2. Modelling Hardware Multiplier and Adder 8

3. Pipelining an FIR filter 8

4. Pipelining IIR filters 11

B. SIGNAL CHARACTERIZATION 12

1. Time Domain Representation 12

2. Frequency Domain representation 13

C. SPECTRUM ESTIMATION 13

1. Least Square Estimator 13

2. Maximum Likelihood(ML) Estimator 16

3. Other Methods 17

D. SELECTION OF ESTIMATION TECHNIQUE 19

1. Model Order Problems 19

2. Choice of the Method 20

III. METHODS for FILTERING 23

A. FILTERING TECHNIQUES 23

1. Filtering via FFT 23
2. Recursive DFT 25
3. Adaptive Filtering 25
B. ADAPTIVE NOTCH FILTERS 28
1. Second-Order Cascaded Notch Filters 28
2. Second-Order Cascaded Signal Canceler 31
a. Computing $\|s(n)\|^2$ 33
C. MULTIPLE NARROWBAND SIGNAL SUPPRESSION 34
1. The Kwan and Martin Filter 34
a. Kwan and Martin Structure 35
b. Calculation of the gradient 36
2. The New Structure 39
IV. MODELLING and SIMULATION 42
A. MODELLING OF SIGNALS 42
1. Sinusoidal Signals 42
2. Narrow-Band Noise 42
3. Bi-Phase Shift Keying Sequence 43
4. Frequency Shift Keying Sequence 44
B. SIMULATION 46
1. Kwan & Martin Algorithm 46
2. The New Algorithm 48
3. Forgetting Filters 48
4. Stability 48
5. Response for Sinusoidal signal 52
6. Response for Narrow Band Noise 52
7. Tracking FSK Signal 55

8. Suppressing BPSK Interference 55
9. Model order mismatch 55
V. HARDWARE IMPLEMENTATION 62
A. HARDWARE FEASIBILITY 62
1. Time Budget 65
2. VLSI Approach 67
VI. CONCLUSIONS 68
A. FUTURE WORK 68
APPENDIX A 69
APPENDIX B 77
REFERENCES 89
INITIAL DISTRIBUTION LIST 91

LIST OF FIGURES

1.1 EW Scenario 2

1.2 A Typical Spectrum 5

2.1 Direct Form II Filter 7

2.2 Pipelined FIR Filter Architecture 10

2.3 Least Square Estimator 15

2.4 Maximum Likelihood Estimator 18

2.5 ARMA(6,1) Spectrum 21

2.6 ARMA(4,1) Spectrum 21

2.7 ARMA(8,1) Spectrum 21

2.8 ARMA(3,2) Spectrum 21

2.9 Spectrum via DFT 22

3.1 Filtering via DFT 24

3.2 Identification - Class I 26

3.3 Inverse Modelling - Class II 26

3.4 Prediction - Class III 27

3.5 Interference Cancelling - Class IV 27

3.6 Cascade of Second-Order Notch Filters 29

3.7 Cascade of Second-Order Cancellation Filters 29

3.8 Higher-Order Notch Filter 30

3.9 Higher-Order Signal Canceller 30

3.10 Sensitivity Function 32

3.11 One Section of Kwan and Martin Filter 32

3.12 Kwan and Martin Structure 37

3.13 The New Structure 40
4.1 Generation of BPSK signal 45
4.2 Kwan and Martin Scheme 47
4.3 The New Scheme 49
4.4 Frequency Response of Notch Filter 51
4.5 Input Spectrum for Sinusoidal Case 53
4.6 Output Spectrum for Sinusoidal Case 53
4.7 Parameter Convergence for Sinusoidal Case 53
4.8 Input Spectrum Narrow-band Case 54
4.9 Output Spectrum Narrow-band Case 54
4.10 Parameter Convergence Narrow-band Case 54
4.11 Input Spectrum FSK Case 56
4.12 Output Spectrum FSK Case 56
4.13 Tracking FSK Signal 56
4.14 Input Spectrum of Broad-band Signal With Interferers 57
4.15 Output Signal - Interferers Removed 57
4.16 Parameter Convergence 57
4.17 Input Spectrum for m=2 and N=3 58
4.18 Output Spectrum for m=2 and N=3 58
4.19 Parameter Adaptation for m=2 and N=3 58
4.20 Input Spectrum for m=1 and N=3 60
4.21 Output Spectrum for m=1 and N=3 60
4.22 Parameter Adaptation for m=1 and N=3 60
5.1 Hardware Implementation of 2nd Order IIR Filter 63
5.2 Basic Building Block 64
5.3 Hardware for 3-Notch Cancellation 66

ACKNOWLEDGMENT

I would like to express my deep sense of gratitude and respect for my thesis advisor Prof Michael Soderstrand for his valuble guidance through out the pursuit of my work here. It is my pleasant duty to express my thanks to my teacher and co-advisor Prof Herchael Loomis. I will be failing in my duties if I do not thank Bob Limes and Elaine Kodres of ECE VAX computer center.

I would like to thank EW curricular officer and his staff for the continious support provided tc me. I would also like to thank the Government of India, Ministry of Defence and the United States Government for providing me such an excellent opportunity.

Lastly I would like to express my sincere thanks to my wife Lavanya for her support during my stay in this place.

I. INTRODUCTION

A. MOTIVATION

The problem of suppressing narrow-band noise from a broad-band spectrum is of considerable importance in the areas of Electronic Warfare and Anti Submarine Warfare scenarios. In this thesis a workable solution is proposed for the problem created by narrow-band interference. This solution will come in the form of an *adaptive digital notch filter*. However before the details of the solution to the problem are discussed, a brief description of the problem is given

1. Electronic Warfare(EW)

A typical EW scenario is given in Fig 1.1. In this scenario a transmitter transmits information over a long distance. As an Electronic Counter Measure(ECM) this information is coded and transmitted as a Direct Sequence Spread Spectrum (DSSS) signal. Inherently this signal is a broad-band signal. However at the EW receiver there is often narrow-band interference from such things as push to talk systems(PTS) that swamp out the received DSSS signal. Sometimes for reasons of signal security PTS operating frequencies are varying with time. Under conditions such as these the EW receiver cannot function effectively. For proper functioning of the EW receiver we must enhance the received signal by selectively and adaptively suppressing these narrow-band signals.

2. Assumptions

This thesis addresses the problem arising at the tactical data communication link. In this EW scenario as dipicted in Fig 1.1 we are attempting to perform signal analysis on a DSSS signal using a EW reciever. This EW receiver is not the des-

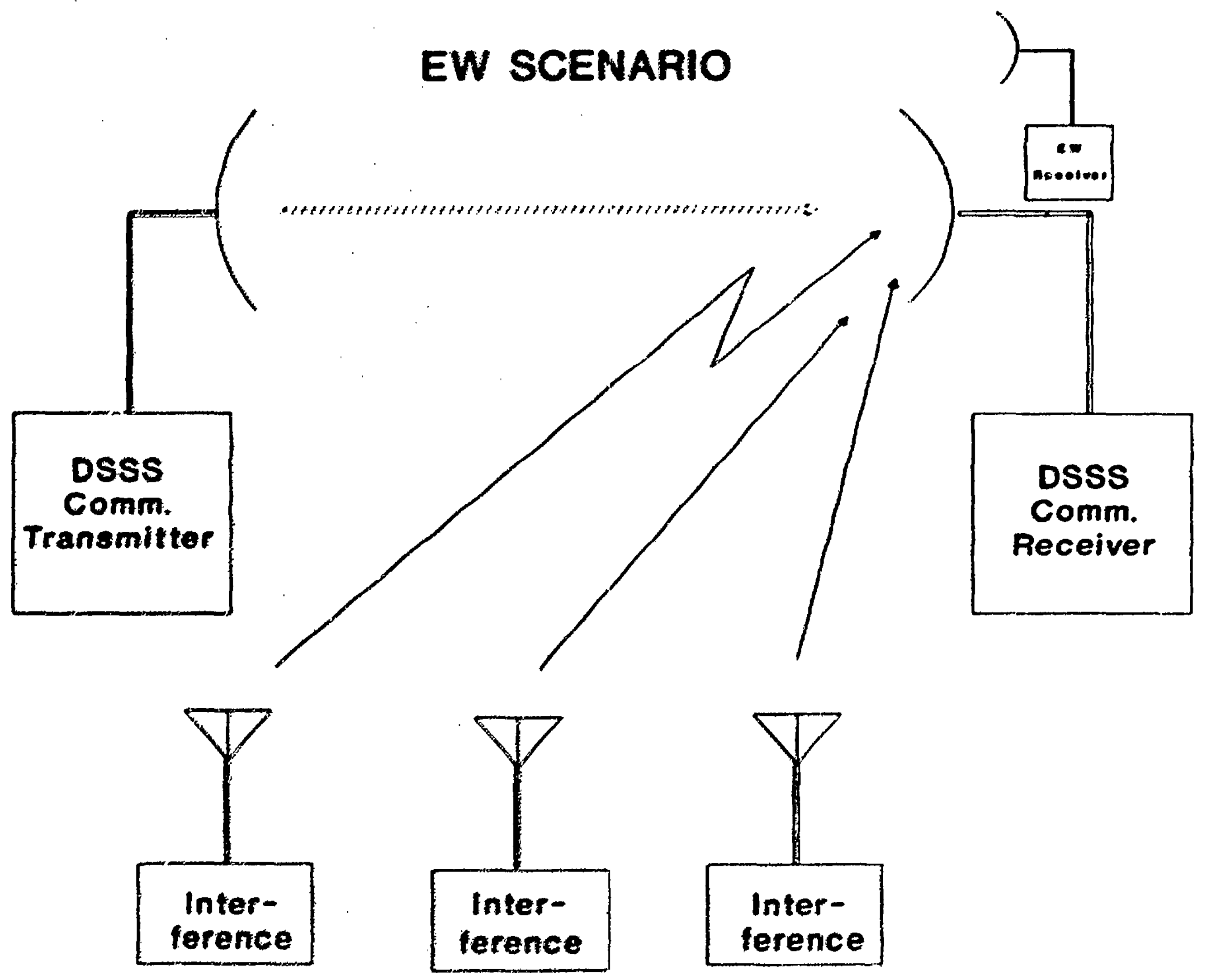

EW SCENARIO
EW Receiver
DSSS Comm. Transmitter
DSSS Comm. Receiver
Inter-ference
Inter-ference
Inter-ference

Fig. 1.1

ignated receiver and hence does not have the spreading code available. For this reason the narrow-band interference severely limits the ability to determine the charaterstics of the recieved signal such as carrier frequency, chip-frequency etc.

A typical DSSS system is the Tactical Information Distribution Systems (JTIDS)[Ref. 7]. Data on such a link is generally in the form of digital messages. A typical L-band (960-1215 Mhz) JTIDS allows transfer of digital data between any properly equipped users within line-of-sight. The typical RF band-width required is around 10Mhz.

Assuming a *Superheterodyne* EW-receiver, the band-width of the Intermediate Frequency (IF) need be only 10Mhz. This enables us to digitize the signal at the base-band with a (20-50)Mhz Analog to Digital(A/D) converter. It is assumed that this signal is digitized and converted into a floating point number. In this thesis it is assumed that the floating point arithmetic is of sufficient precision that rounding errors are not a significant problem.

B. PROBLEM DEFINITION

The signal described in section (I.A.3) can be thought of as a broad-band signal with additive narrow-band signals occurring at arbitrary frequencies and times. These narrow-band signals can be modeled as either pure sinusoidal signals, as outputs of a narrow-band filter excited by white noise, or as any band-limited narrow-band signal.

We shall assume that the signal y_k is received by the receiver and is defined as

$$y_k = w_k + \sum_{i=1}^{3} x_k^i + \gamma_k \tag{1.1}$$

where x_k^i, the interferer, is either a pure sinusoid or a narrow-band signal whose frequency (i.e center frequency) can vary slowly with time. The term w_k is the broad-band DSSS signal of interest and γ_k is White Gaussian Noise(WGN) with zero mean and variance σ^2 $N(0, \sigma^2)$ The goal is to remove only the narrow-band interferers and obtain the output $(w_k + \gamma_k)$ for further processing. A typical spectrum for y_k is given in Fig 1.2.

In this thesis, the design for an adaptive digital filter that will achieve the goal of removing narrow-band interferers from a broad-band signal of interest is developed. This thesis is organized into five chapters. chapter I is the introduction. The chapter II concentrates on building necessary background on such things as the Least Square Estimator and Maximum-Likelihood Estimator. Chapter III briefly explains existing methods of adaptive filtering with emphasis on adaptive notch filtering techniques . The new algorithm [Ref. 26] developed for the purpose of addressing the current problem of interest is also explained. Chapter IV is totally devoted to modelling various signals required for testing. Simulation results are discussed and presented. Chapter V provides a brief survey of existing hardware components and a look at possible hardware structures and their limitations. Chapter VI is conclusions.

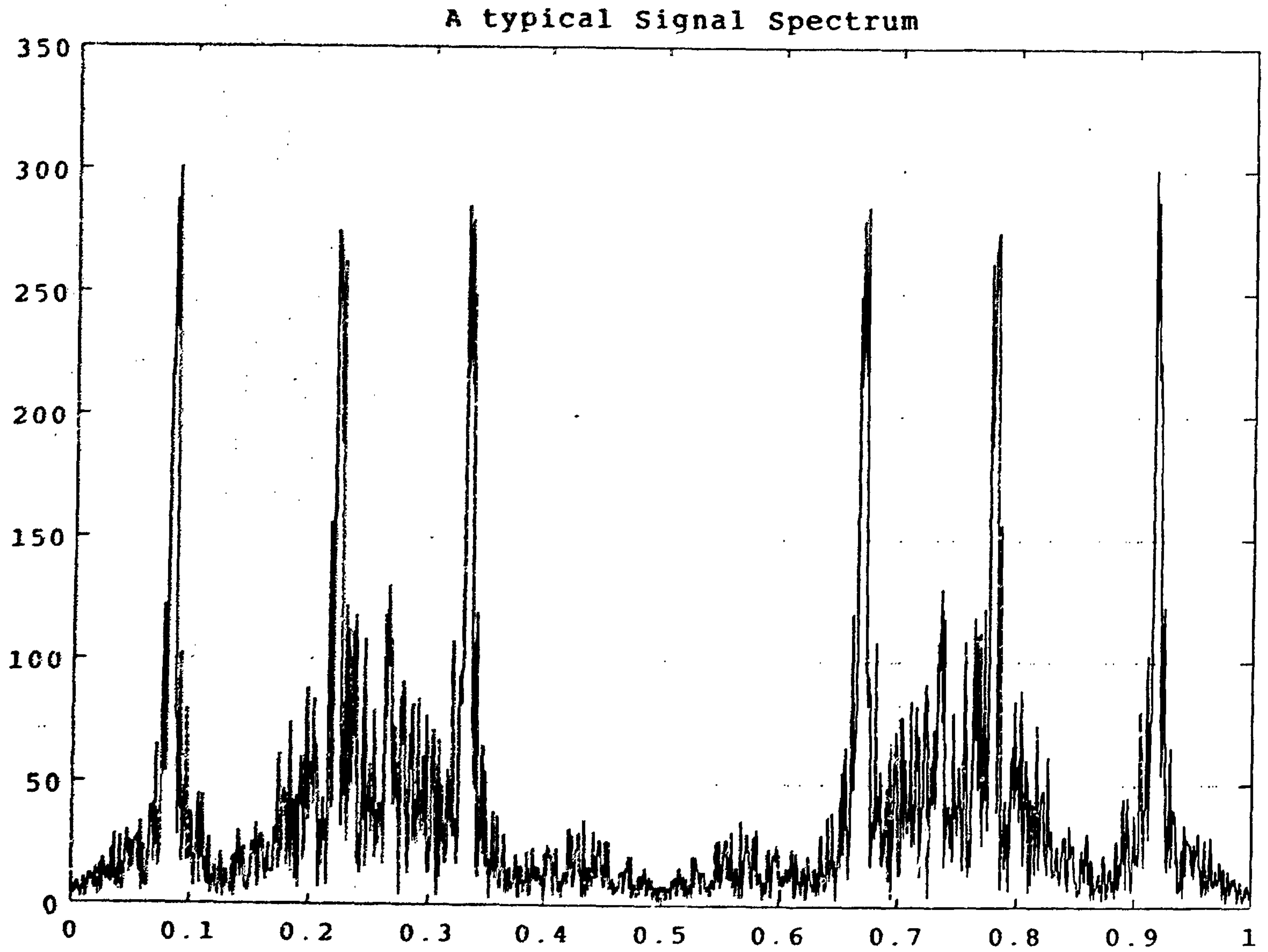

A typical Signal Spectrum
350
300
250
200
150
100
50
0
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

Fig 1.2

II. ESTIMATION TECHNIQUES

A. DIGITAL FILTERS

Digital Filters can be characterized by their impulse response, their transfer function, or by a difference equation [Ref. 13] . A typical Infinite-Impulse-Response(IIR) filter represented in difference equation form is given by

$$x_k = a_1 x_{k-1} + a_2 x_{k-2} + a_3 x_{k-3} + b_1 u_k + b_2 u_{k-1} \tag{2.1}$$

while an Finite Impulse Response(FIR) filter is given by

$$x_k = b_1 u_k + b_2 u_{k-1} + b_3 u_{k-3} \tag{2.2}$$

where u_k is the input to the filter and x_k is the output of the filter. x_k and u_k are the sampled values of the continuous functions $x(t)$ and $u(t)$. By choosing T_s as the sampling time we notionally write the discrete signal as

$$x_k = x(k) = x(kT_s) \tag{2.3}$$

Let $X(\Omega)$ be the spectrum of the signal x_k, where Ω is the angular frequency component present in the actual signal. For analysis purposes consider normalized frequency given as $\omega = \Omega T_s$. With this definition we do not have to use the actual frequencies but only the normalized frequencies. Assuming that there is no aliasing while sampling the maximum normalized frequency is 0.5 cycle/sample or π rad/sample.

1. Some Hardware Schemes

Fig 2.1 represents a conventional Direct Form II representation of a digital filter [Ref. 18]. However for high-speed throughput it is often necessary to have a pipelined architecture. FIR filters are highly amenable for pipelining thus giving a high throughput rate for computations.

Direct Form II

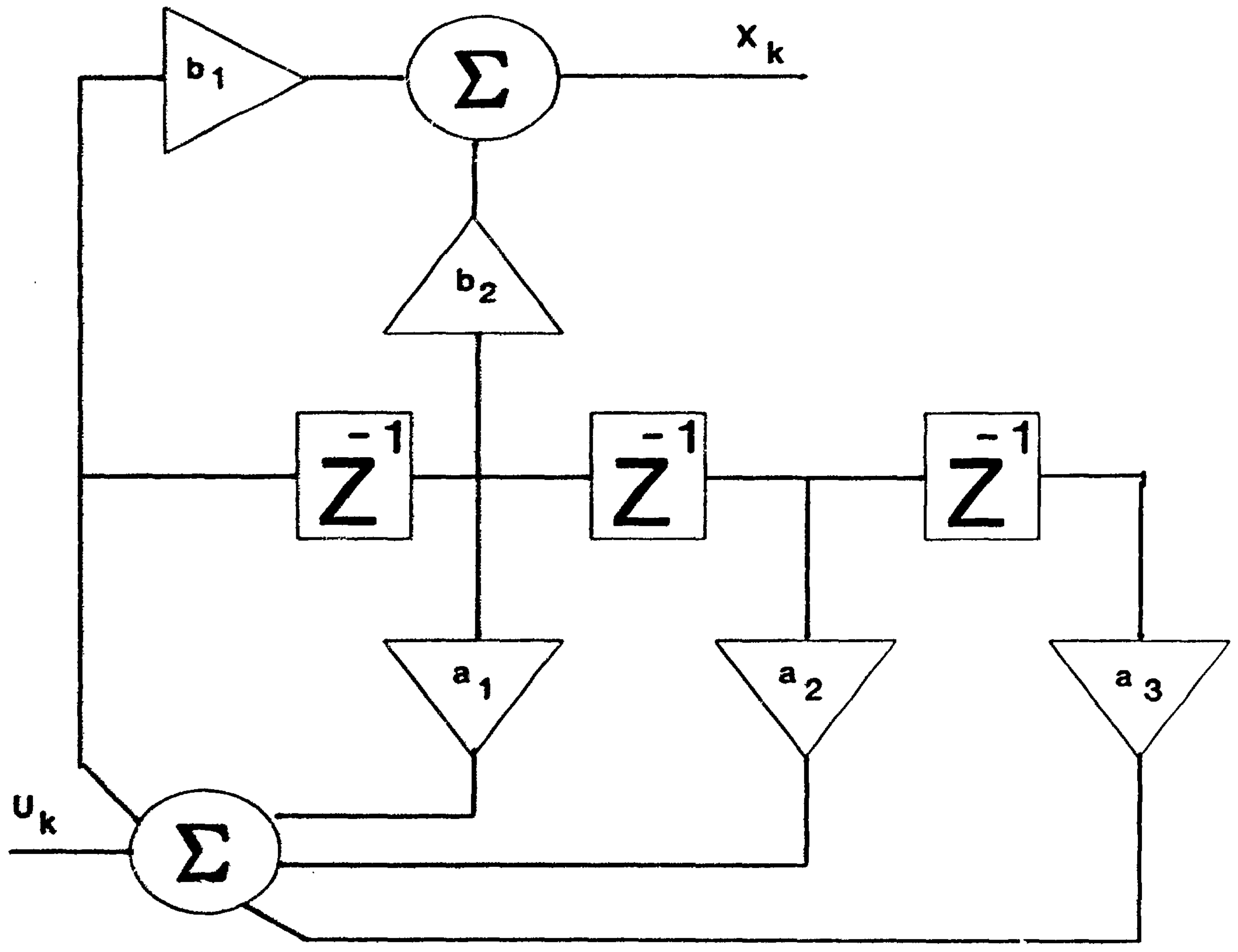


Fig 2.1

2. Modelling Hardware Multiplier and Adder

Multiplication of two variables say b_1 and u_k in hardware can be modelled as

$$b_1 \hat{*} u_k = b_1 * u_k D_1 = b_1 u_k D_1 \tag{2.4}$$

where $\hat{*}$ is the hardware multiplier and $*$ is the *ideal* multiplier and D_1 is the propagation delay associated with the multiplication. Similarly a hardware adder is designated as $\hat{+}$ while the *ideal* adder is given as $+$ and the corresponding delay associated with the adder is D_2. Then an addition of two variables u_k and u_{k-1} can be written as

$$u_k \hat{+} u_{k-1} = (u_k + u_{k-1}) D_2 \tag{2.5}$$

Each of the previous devices can be incorporated into a pipeline digital filter structure by following it with a synchronizing register. The clock period must then be less than $D + t_s + t_r$ where t_s is the register setup time and t_r the register propagation delay.

3. Pipelining an FIR filter

In this section we wish to realize

$$x_k = b_1 * u_k + b_2 * u_{k-1} + b_3 * u_{k-2} \tag{2.6}$$

It is acceptable to have an overall delay D^N , However it is desirable to minimize N:

$$\hat{x}_k = D^N \{b_1 * u_k + b_2 * u_{k-1} + b_3 * u_{k-2}\} \tag{2.7}$$

First associate one D with each multiplier and form real multipliers by the notation $D* = \hat{*}$. Then we get

$$\hat{x}_k = D^{N-1} \{b_1 D * u_k + b_2 D * uk - 1 + b_3 D * u_{k-2}\} \tag{2.8}$$

Now form the real multiplier

$$\hat{x}_k = D^{N-1} \{b_1 \hat{*} u_k + b_2 \hat{*} uk - 1 + b_3 \hat{*} u_{k-2}\} \tag{2.9}$$

Now we use one additional D to associate with the adding of $b_2\hat{*}u_{k-1}$ and $b_3\hat{*}u_{k-2}$. (Note: It is important to choose the most delayed values first in order to minimize N)

$$\hat{x}_k = D^{N-2}\{Db_1\hat{*}u_k + D(b_2\hat{*}u_{k-1} + b_3\hat{*}u_{k-2})\} \tag{2.10}$$

Now form the real adder

$$\hat{x}_k = D^{N-2}\left\{Db_1\hat{*}u_k + (b_2\hat{*}u_{k-1}\hat{+}b_3\hat{*}u_{k-2})\right\} \tag{2.11}$$

Now we use an additional delay D to associate with the final adder to get a real adder:

$$\hat{x}_k = D^{N-3}\left\{Db_1\hat{*}u_k\hat{+}(b_2\hat{*}u_{k-1}\hat{+}b_3\hat{*}u_{k-2})\right\} \tag{2.12}$$

Finally we choose the system delay such that $D = z^{-1}$ and replace $u_{k-2} = z^{-2}u_k$ and $u_{k-1} = z^{-1}u_k$, thus we get

$$\hat{x}_k = z^{-N+3}\left\{z^{-1}b_1\hat{*}u_k\hat{+}(z^{-1}b_2\hat{*}u_k\hat{+}z^{-2}b_3\hat{*}u_k)\right\} \tag{2.13}$$

Now factor z^{-1} out of the expression to obtain

$$\hat{x}_k = z^{-N+2}\{b_1\hat{*}u_k\hat{+}(b_2\hat{*}u_k\hat{+}z^{-1}b_3\hat{*}u_k)\} \tag{2.14}$$

Note that $N = 2$ is the minimum possible value of N for a causal filter. Choose $N = 2$ then

$$\hat{x}_k = \{b_1\hat{*}u_k\hat{+}(b_2\hat{*}u_k\hat{+}z^{-1}b_3\hat{*}u_k)\} \tag{2.15}$$

Equation (2.15) is very convenient for pipelining and is given *one to one* in Fig 2.2.

The above procedure is general for any FIR filter. Pipelining an FIR filter results in maximizing the sampling or throughput frequency at the expense of a delay or latency in the availability of the output.

Pipelined FIR Filter Architecture

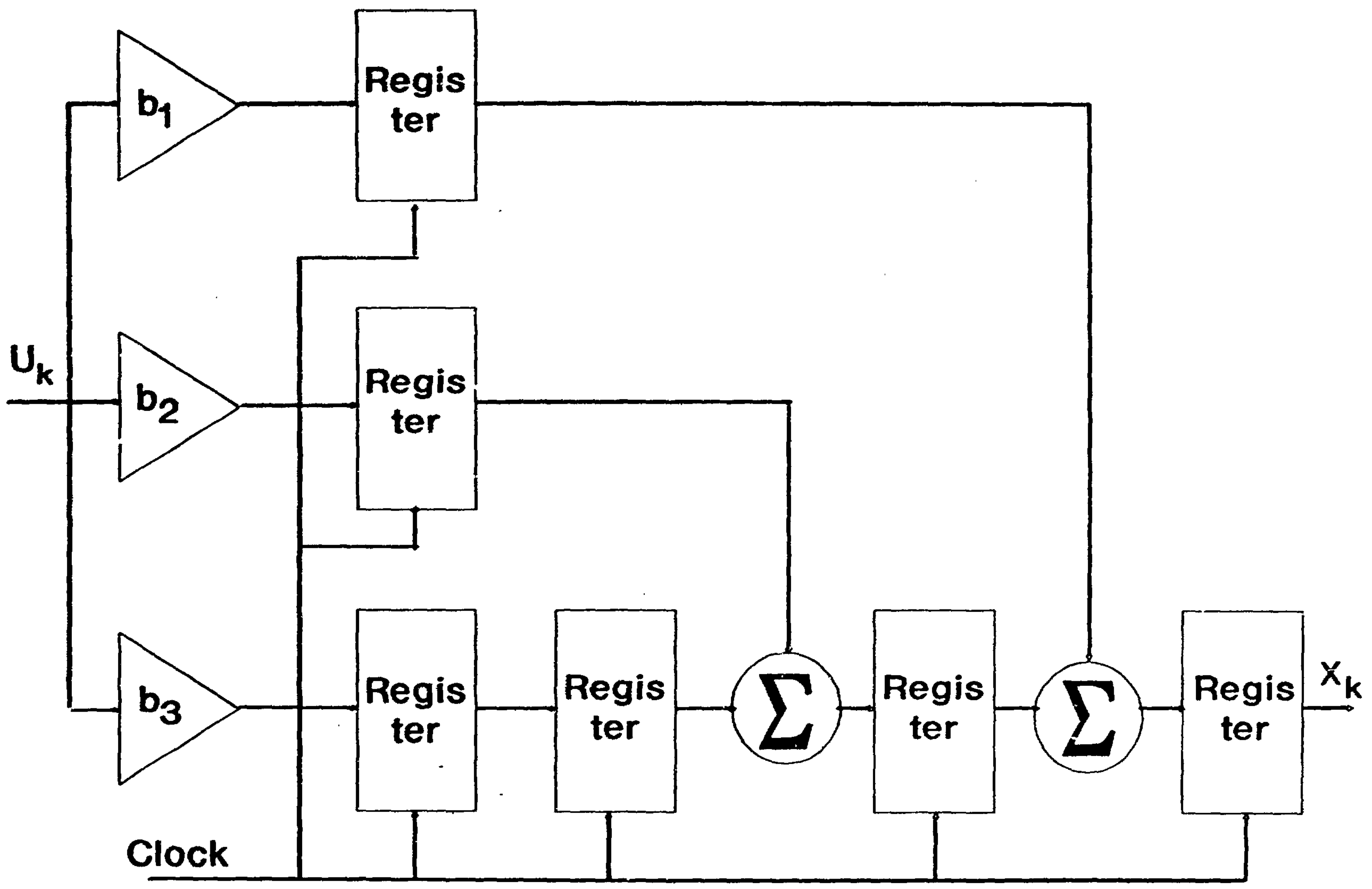


Fig. 2.2

4. Pipelining IIR filters

Pipelining an IIR filter is considerably more difficult than an FIR filter. This is mainly because the delay in the availability of the output makes it impossible to feed back output values with short delay as is required for the straightforward IIR implementation. As an example consider a simple 1st order IIR filter

$$x_k = a_1 * x_{k-1} + u_k \tag{2.16}$$

As in the FIR case we introduce a delay D^N as follows.

$$\hat{x}_k = D^N(a_1 * x_{k-1} + u_k) \tag{2.17}$$

Now we use one delay to form the real multiplier:

$$\begin{aligned} \hat{x}_k &= D^{N-1}(a_1 D * x_{k-1} + Du_k) && (2.18) \\ &= D^{N-1}(a_1 \hat{*} x_{k-1} + Du_k) && (2.19) \end{aligned}$$

We use one more D to form the real adder

$$\hat{x}_k = D^{N-2}(a_1 D * x_{k-1} \hat{+} u_k) \tag{2.20}$$

Finally assume $D = z^{-1}$ and $x_{k-1} = z^{-1} x_k$ then

$$\begin{aligned} \hat{x}_k &= z^{-N+2}(a_1 \hat{*} z^{-1} x_k \hat{+} z^{-1} u_k) && (2.21) \\ &= z^{-N+1}(a_1 \hat{*} x_k \hat{+} u_k) && (2.22) \end{aligned}$$

For $\hat{x}_k = x_k$, N must be zero. However $N \geq 1$ is required for a causal filter.

Solutions to this problem exist. They use a higher order difference equation representation of the filter with equivalent characteristics, so that feedback loop delay greater than 1 can be tolerated. More details can be found in the literature [Ref. 14] regarding pipelining of IIR digital filters.

B. SIGNAL CHARACTERIZATION

The signal y_k of equation (1.1) can be characterized either in the *time domain* or *frequency domain* and the signal can be easily transformed from one domain to the other via a linear transformation.

1. Time Domain Representation

In the time domain representation, the signal y_k is modelled as the output of a linear system which may be either an all pole or a pole-zero system [Ref. 13]. The input is assumed to be white noise but this input to the system is not accessible and is only conceptual in nature. Let the system under consideration be

$$y_k = \sum_{i=1}^{p} a_i y_{k-i} + \sum_{j=0}^{q} b_{j+1} \gamma_{k-j} \tag{2.23}$$

where γ_k is a white noise and y_k is the output of the system. Equation (2.23) also can be represented as a Transfer Function(TF) [Ref. 17]

$$H(z) = \frac{B(z)}{A(z)} \tag{2.24}$$

or

$$x_k = \left\{ \frac{B(z)}{A(z)} \right\} \gamma_k \tag{2.25}$$

where

$$A(z) = 1 - \sum_{i=1}^{p} a_i z^{-i} \tag{2.26}$$

and

$$B(z) = \sum_{j=0}^{q} b_{j+1} z^{-j} \tag{2.27}$$

we define the parameter vector as

$$\mathbf{p}^t = [a_1, a_2, \cdots, a_p, b_1, b_2, \cdots, b_q] \tag{2.28}$$

The parameter $\mathbf{p}$ completely characterizes the signal x_k The above system defined by the equation (2.23) is also known as an Auto-Regressive-Moving-Average(ARMA) model [Ref. 12],[Ref. 15]. It is conventional to denote a p poles and q zeros system as ARMA(p,q).

2. Frequency Domain representation

The signal y_k of equation (2.23) can also be represented as a vector

$$\mathbf{Y}_N = [y_1 \cdots y_N] \tag{2.29}$$

which can be transformed into the frequency domain by the DFT relation

$$g_k = \frac{1}{N}\left\{\sum_{n=0}^{N-1}\left(W^{nk}y_n\right)\right\} \tag{2.30}$$

where $W = e^{-\frac{j2\pi}{N}}$ and $j = \sqrt{-1}$. In general g_k is a complex quantity. However we restrict our interest to the power spectrum a real quantity given as

$$s_k = g_k g_k^* \tag{2.31}$$

The series S_N

$$\mathbf{S}_N = [s_1 \cdots s_N] \tag{2.32}$$

is another form of representing the signal and is designated as the *power spectrum* of the signal x_k. Even though the signal in this domain is very convenient to handle, computational complexity and other considerations limit its use for online application. Also in this representation *phase* information of the signal is lost.

C. SPECTRUM ESTIMATION

1. Least Square Estimator

Consider the signal y_k the output of a linear system excited by a white noise as given by the equation (2.23). Now our problem is to estimate the parameter vector p by obtaining successive measurements of y_k. We shall define the vector

$$\mathbf{x}_k^t = [y_{k-1}, \cdots, y_{k-p}, \gamma_k, \cdots, \gamma_{k-q}] \tag{2.33}$$

and $z_k = y_k$ so that the difference equation (2.23) can be rewritten as

$$z_k = \mathbf{p}_k^t \mathbf{x}_k \tag{2.34}$$

an elegant solution [Ref. 2] for the above problem can be written as

$$\hat{\mathbf{p}}_k = \hat{\mathbf{p}}_{k-1} + \mathbf{H}_k \mathbf{x}_k e_k \tag{2.35}$$

where

$$\mathbf{H}_k = \left(\sum_{i=1}^{k} \mathbf{x}_i \mathbf{x}_i^t \right)^{-1} \tag{2.36}$$

and

$$e_k = z_k - \hat{\mathbf{p}}_{k-1}^t \mathbf{x}_k \tag{2.37}$$

The block solution for the equation (2.34) can be also written in the form

$$\mathbf{p}_k = \mathbf{H}_k \left(\sum_{i=1}^{k} \mathbf{x}_k z_k \right) \tag{2.38}$$

Equation (2.37) can also viewed as $\hat{A}x_k - \hat{B}\gamma_k$ and this is represented in Fig 2.3.

Computing the matrix $\mathbf{H}_k$ via equation (2.36) is computationally inefficient and a recursive solution via the matrix inversion lema [Ref. 3] is preferred and is given by

$$\mathbf{H}_k = \mathbf{H}_{k-1} - \left[\frac{\mathbf{H}_{k-1} \mathbf{x}_k \mathbf{x}_k^t \mathbf{H}_{k-1}}{1 + \mathbf{x}_k^t \mathbf{H}_{k-1} \mathbf{x}_k} \right] \tag{2.39}$$

Staticians refer to the $\mathbf{H}_k$ matrix as the *covariance* matrix while optimization people refer to $\mathbf{H}_k$ as the *Hessian* of the objective function, where the objective function is given by

$$J_k = \sum_{i=1}^{k} e_i^2 \tag{2.40}$$

Appendix A gives a computer program based on this approach that was used for this thesis work to identify system parameters via Equations (2.35), (2.36), (2.37), (2.39). The above method is known to work well as long as the measured value of the state of the filter is free from noise. In the event z_k is not noise free, estimates are known to be biased.

Least Square Estimator

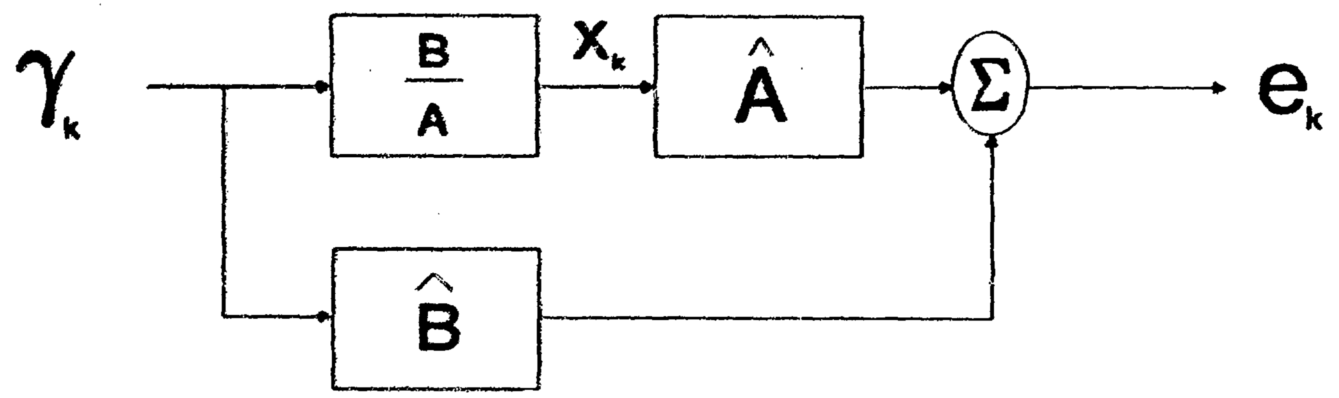


Fig. 2.3

2. Maximum Likelihood(ML) Estimator

The ML estimator for the above problem is conventionally obtained by considering a log-likelihood function [Ref. 16] and minimizing it. However for getting a better physical understanding the approach given by *Young*[Ref. 2] will be followed. Consider a system as

$$y_k = \left[\frac{B(z)}{A(z)}\right]\gamma_k \tag{2.41}$$

and an auxiliary system referred to as the *model* is given as

$$\hat{x}_k = \left[\frac{\hat{B}(z)}{\hat{A}(z)}\right]\gamma_k \tag{2.42}$$

and we define the function to be minimized as $L = e_k^2(\mathrm{p})$ where $e_k = y_k - \hat{x}_k$. To minimize this function the derivatives and the Hessian of the function L where $L = (y_k - \hat{x}_k)^2$ need to be obtained. First take partial derivative of L with respect to each parameter. For example

$$\frac{\partial L}{\partial b_1} = 2(y_k - \hat{x}_k)(-\frac{\partial \hat{x}_k}{\partial b_1}) = -2e_k(\frac{\partial \hat{x}_k}{\partial b_1}) \tag{2.43}$$

The partial $\frac{\partial \hat{x}_k}{\partial b_1}$ can be obtained by differentiating equation(2.42) as

$$\frac{\partial}{\partial b_1}\left\{\frac{\hat{B}(z)}{\hat{A}(z)}\gamma_k\right\} = \frac{\partial}{\partial b_1}\left(\left[\frac{b_1 + b_2 z^{-1}}{1 - a_1 z^{-1} - a_2 z^{-2} - a_3 z^{-3}}\right]\gamma_k\right) \tag{2.44}$$

$$= \left[\frac{1}{\hat{A}(z)}\right]\gamma_k \tag{2.45}$$

Similarly the other partial $\frac{\partial \hat{x}_k}{\partial a_i}$ can be obtained as

$$\frac{\partial}{\partial a_1}\left\{\frac{\hat{B}(z)}{\hat{A}(z)}\gamma_k\right\} = \left(\frac{\hat{A}(z)\frac{\partial}{\partial a_1}\{\hat{B}(z)\} - \hat{B}(z)z^{-1}}{[\hat{A}(z)]^2}\right)\gamma_k \tag{2.46}$$

$$= -\left(\frac{\hat{B}(z)}{[\hat{A}(z)]^2}z^{-1}\right)\gamma_k \tag{2.47}$$

since $\frac{\partial}{\partial a_1}\{B(z)\} = 0$ as $\hat{B}(z)$ has no terms containing a_1. Using equations (2.45) and (2.47) the partials are:

$$\begin{aligned} \frac{\partial \hat{x}_k}{\partial b_1} &= u_k^* = \left[\frac{1}{\hat{A}(z)}\right]\gamma_k && (2.48)\\ \frac{\partial \hat{x}_k}{\partial a_1} &= x_k^* = \left[\frac{\hat{B}(z)}{[\hat{A}(z)]^2}\right]\gamma_k && (2.49)\\ \frac{\partial y_k}{\partial a_1} &= y_k^* = \left[\frac{1}{\hat{A}(z)}\right]y_k && (2.50)\\ & && (2.51) \end{aligned}$$

by defining

$$\begin{aligned} \mathbf{p}_k^t &= [a_1, a_2, a_3, b_1, b_2] && (2.52)\\ \mathbf{x}_k^t &= [x_{k-1}^*, x_{k-2}^*, x_{k-3}^*, u_k^*, u_{k-1}^*] && (2.53)\\ \mathbf{z}_k^t &= [y_{k-1}^*, y_{k-2}^*, y_{k-3}^*, u_k^*, u_{k-1}^*] && (2.54) \end{aligned}$$

the Hessian may be approximated by:

$$\mathbf{H}_k = \left(\sum_{i=1}^{k} \mathbf{x}_i \mathbf{z}_i^t\right)^{-1} \qquad (2.55)$$

Using the above equations the parameter can be estimated via

$$\mathbf{p}_k = \mathbf{p}_{k-1} + \mu \mathbf{H}_k \mathbf{x}_k e_k \qquad (2.56)$$

where μ is the step size in incrementing the parameter vector. A block schematic for the algorithm was given in Fig 2.4. A serious drawback of this method is the stability while incrementing the parameters.

3. Other Methods

There are many other methods such as the Instrumental Variable (IV) approach, the Approximate-Maximum-Likelihood (AML) Method and a combination method IV-AML available in the literature [Ref. 2].

Maximum Likelyhood Method

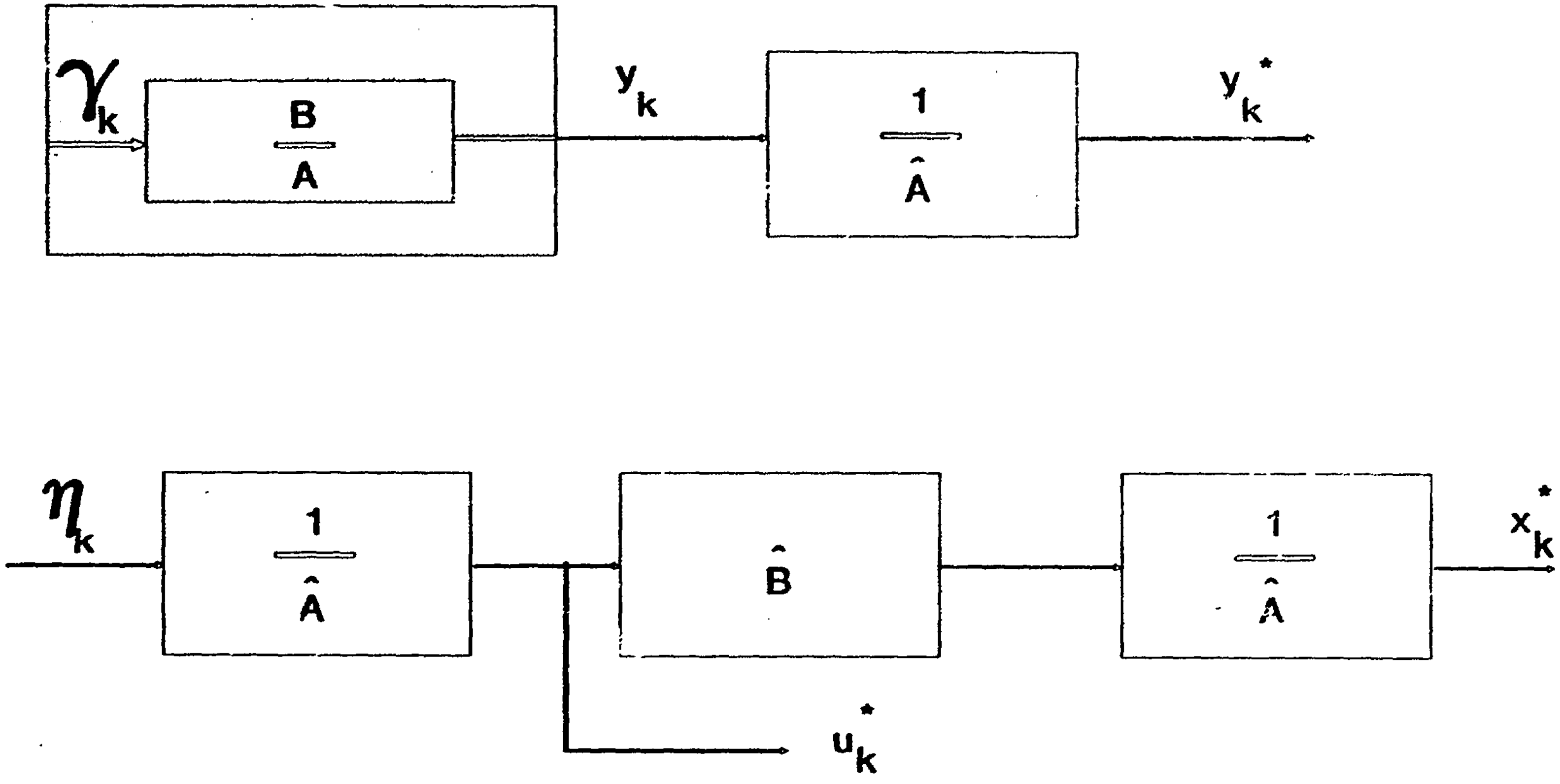


Fig. 2.4

D. SELECTION OF ESTIMATION TECHNIQUE

Before we select the appropriate method for the problem of interest, model-order problems must be considered.

1. Model Order Problems

In the Least-Square Estimator for the parameter vector $\mathbf{p}$ of the signal x_k we need to assume the dimension of $\mathbf{p}$ or the order of the system. Since the system order is often not known a higher-order model estimate is assumed. Now we investigate higher-order model fits for a lower-order data.

Consider the system equation (2.23) with parameter given as

$$\mathbf{p}^t = [1.7, -1.53, 0.648, 1, 0.6] \tag{2.57}$$

For the above system a Random Binary Noise(RBN) was given as the input u_k and the corresponding output x_k is obtained. For the time series u_k and x_k an ARMA(6,1) model was fit using the program given in the appendix and the parameter was estimated as

$$\mathbf{p}^t = [2.26533, -2.79577, 2.17249, -1.09877, 0.446598, -0.110151, 0.998175] \tag{2.58}$$

Fig 2.5 gives the parametric spectrum for the above.

For the same data an ARMA(4,1) model was used and generating a parameter vector

$$\mathbf{p}^t = [2.16988, -2.41286, 1.47436, -0.365412, 0.999317] \tag{2.59}$$

Fig 2.6 gives the parametric spectrum for the above values.

Lastly for the same data an ARMA(8,1) model was used and generating a parameter vector

$$\mathbf{p}^t = [2.276, -2.835, 2.267, -1.282, 0.703, -0.353, 0.143, -0.0035, 0.9999] \tag{2.60}$$

Fig 2.7 gives the corresponding spectrum.

It is interesting to compare these with the actual parametric spectrum given in Fig 2.8 For the sake of completion and comparison, the spectrum given in Fig 2.9 of the output time series x_k computed via a DFT program is also included. This simulation demonstrates that the existence of multiple solutions, hence an important step in the estimation procedure is to choose an appropriate model order.

2. Choice of the Method

Since the filter hardware must be implemented in real-time at the frequencies of 10 to 20 Mhz computational complexities must be kept to a minimum. These methods although providing good performance, are not well suited to hardware implementations. This motivates looking into new filtering methods.

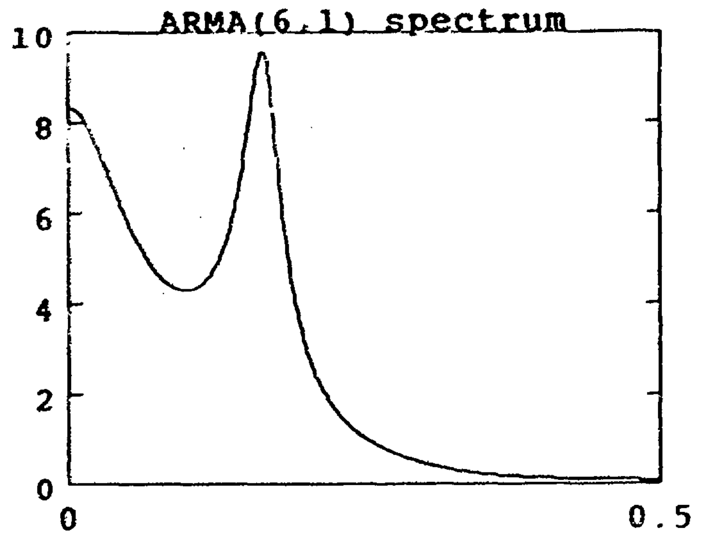


Fig 2.5

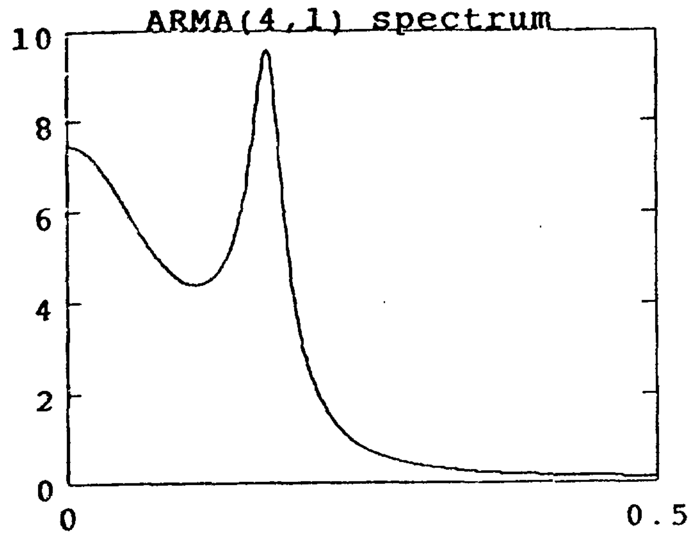


Fig 2.6

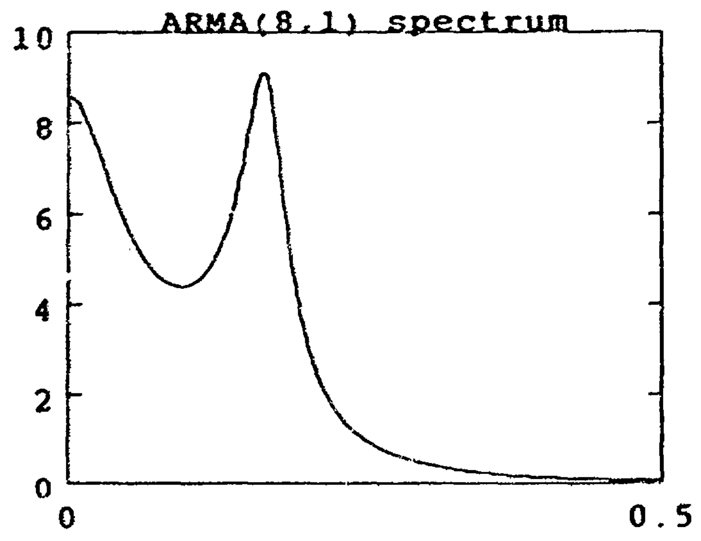


Fig 2.7

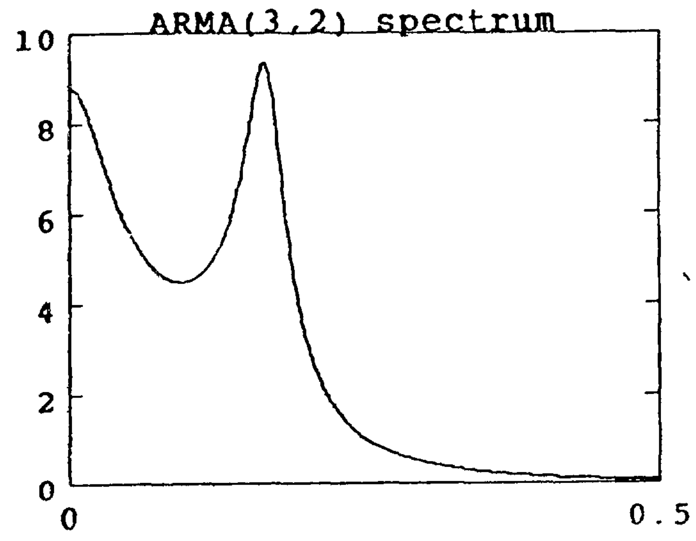


Fig 2.8

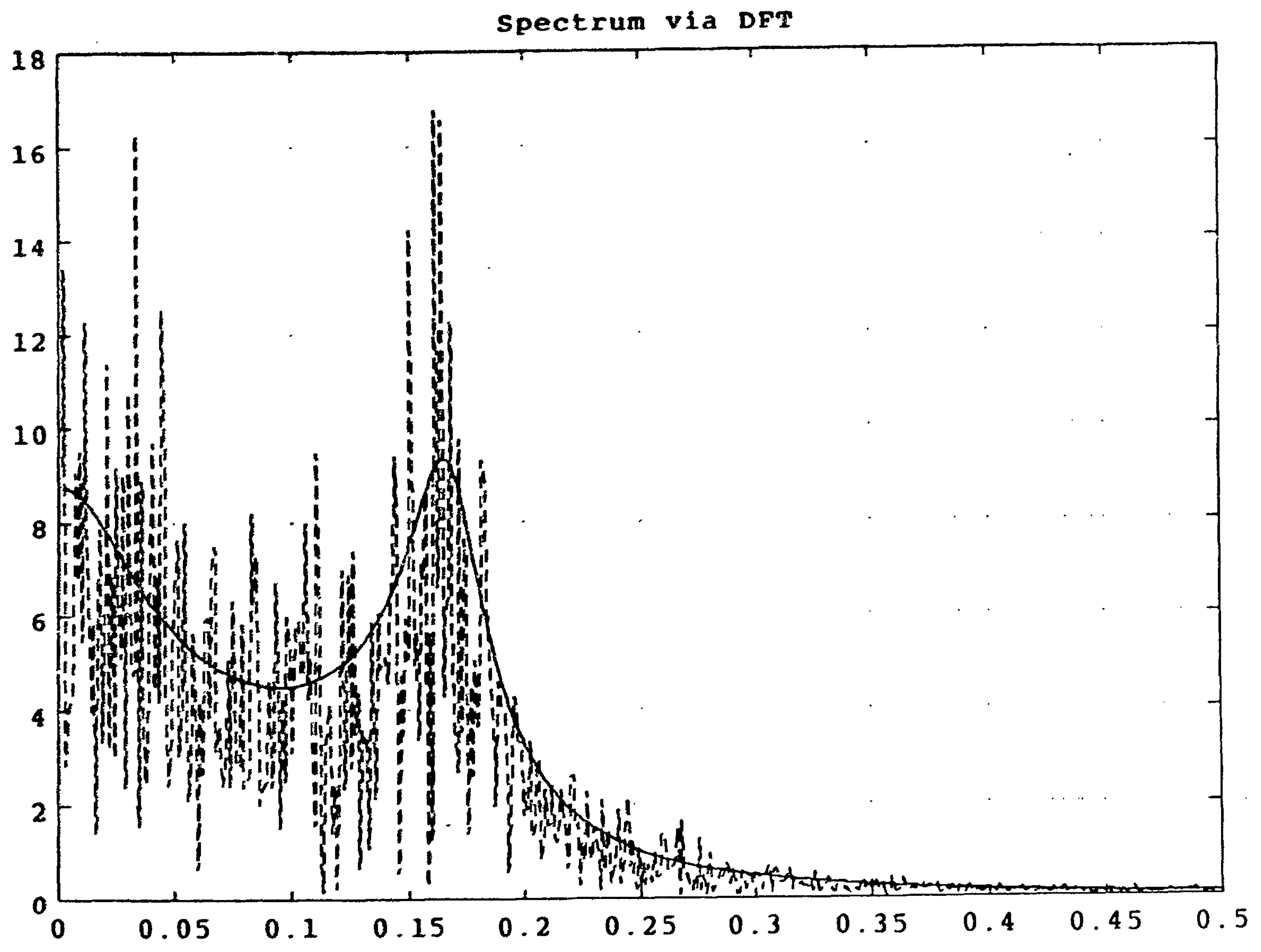

Spectrum via DFT
18
16
14
12
10
8
6
4
2
0
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
0.4
0.45
0.5

Fig 2.9

III. METHODS for FILTERING

A. FILTERING TECHNIQUES

Filtering in a broad sense is selectively suppressing a portion of the spectrum of the given signal. In this chapter several filtering techniques are explored in the light of the narrow-band interference problem in order to identify applicable filtering techniques.

1. Filtering via FFT

Any given signal can be conveniently transformed into its power spectrum via equation 2.31. Assuming the spectrum of the desired filter to be defined by w_k. The weighted output is given by:

$$v_k = s_k w_k \tag{3.1}$$

and the corresponding filtered output y_k is obtained by taking the inverse DFT of the signal v_k. A simple block schematic representing this idea is given in Fig 3.1. Details of this approach are available in various references [Ref. 22].

Fast algorithms such as the FFT for computing the DFT make it possible to do the above process in real time by using dedicated hardware. Honeywell makes the HDSP66110 and HDSP66210 Digital Signal Processing chip pair which are ideally suited for these applications. This chip pair can perform a single complex multiply in 40ns [Ref. 28]. However the problem of filtering adaptively [Ref. 10] (*i.e* varying w_k according to a criterion) demands more computing power which can be obtained by augmenting the DSP chip pair with a processor like the R3000 which has a Reduced Instruction Set Computer(RISC) architecture with a high instruction execution rate of approximately 25 Million Instructions Per Second(MIPS) [Ref. 27].

Filtering via FFT

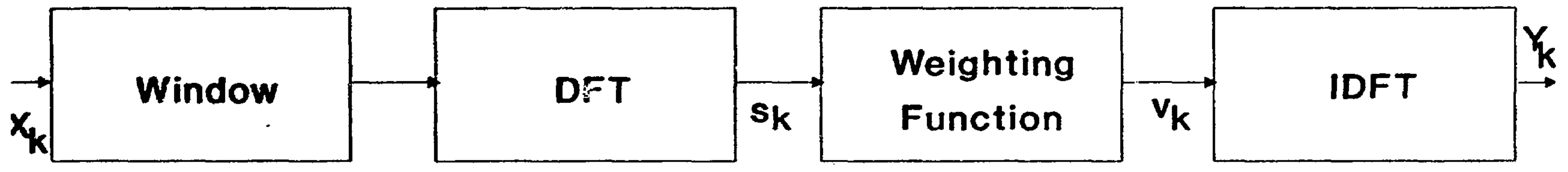


Fig. 3.1

2. Recursive DFT

Implementation of a higher order FIR filter using a recursive DFT [Ref. 23],[Ref. 24] is also very convenient for eliminating narrowband interference. The basic concept is that the FIR filter is expressed as a product of two filter sections. One section is a filter with its zeros being equally spaced on the unit circle. This is achieved by a delay. The second section is a pole-producing section. Pole-zero cancellation results in the desired FIR filter. More details can be seen in the references [Ref. 23],[Ref. 24].

3. Adaptive Filtering

Adaptive filters can be placed into four classes based upon the choice of the *training sequence* and the reference model for adaptation. *Simon* [Ref. 21]has classified the *Adaptive Filters* into the following four classes

- Identification - Class I
- Inverse Modelling - Class II
- Prediction - Class III
- Interference Canceling - Class IV

Fig [3.2 - 3.5] give the block schematics of the various classes of adaptive filters. In our problem, the *reference model* is naturally a *narrow-band bandpass* filter since the interference signal is a narrow-band signal. The class of filtering that best suits our problem is a combination of Class III and Class IV type of filter [Fig 3.4 and 3.5]. This logically points to an adaptive notch filter. Due to *apriori* knowledge of the interference signal, only the *parametric* approach was. However the DFT-based techniques may also offer a good solution and should be the subject of future investigations.

Identification (Class 1)

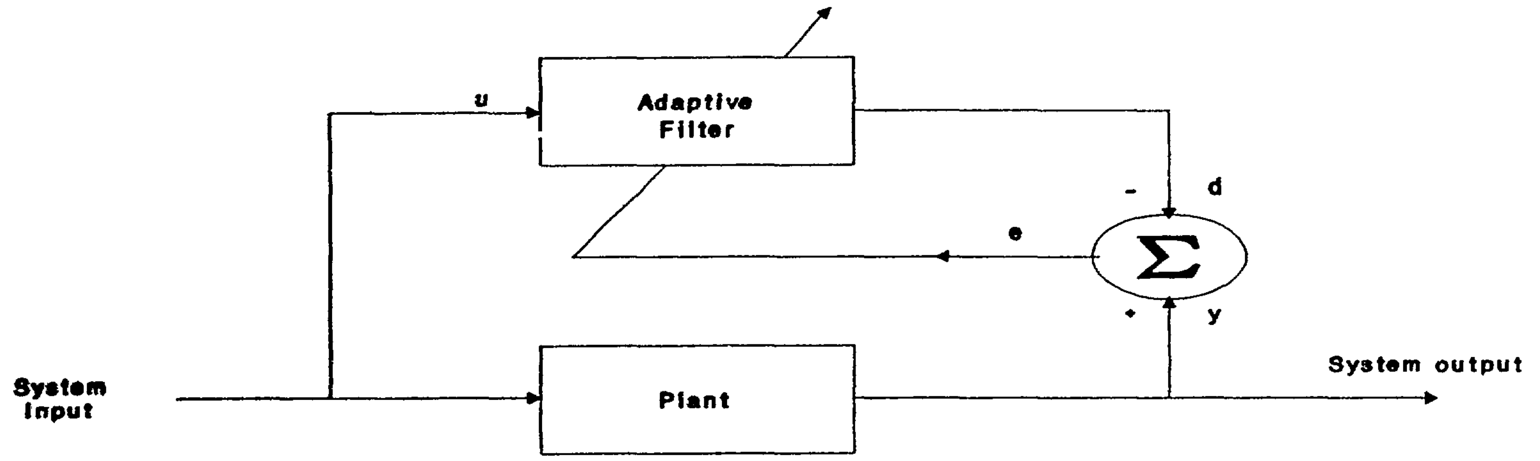


Fig. 3.2

Inverse Modelling (Class II)

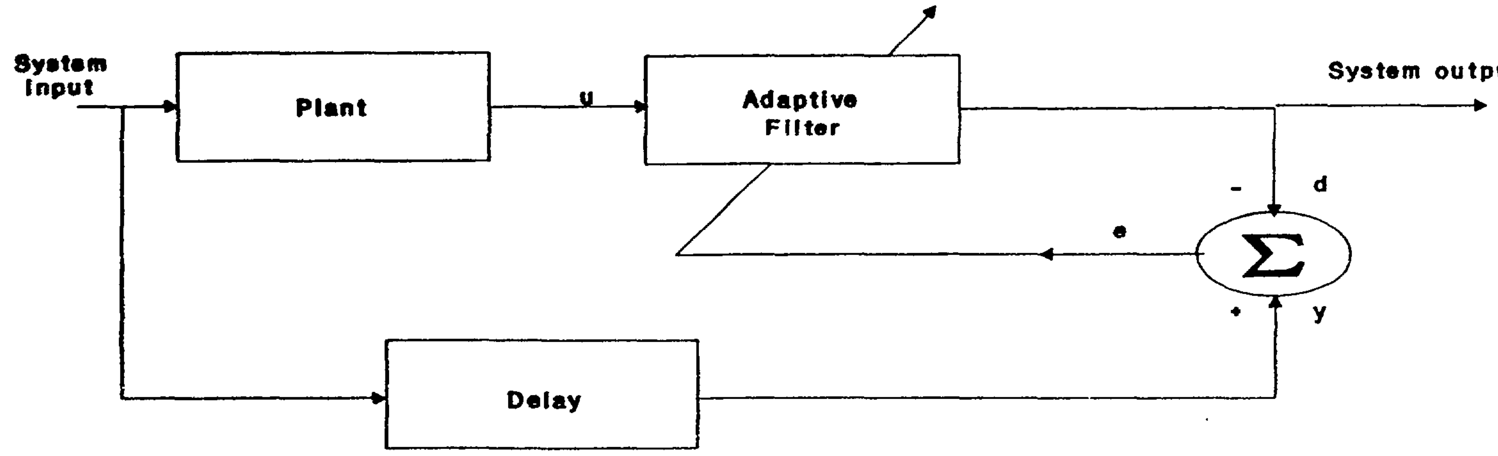


Fig. 3.3

Prediction (Class III)

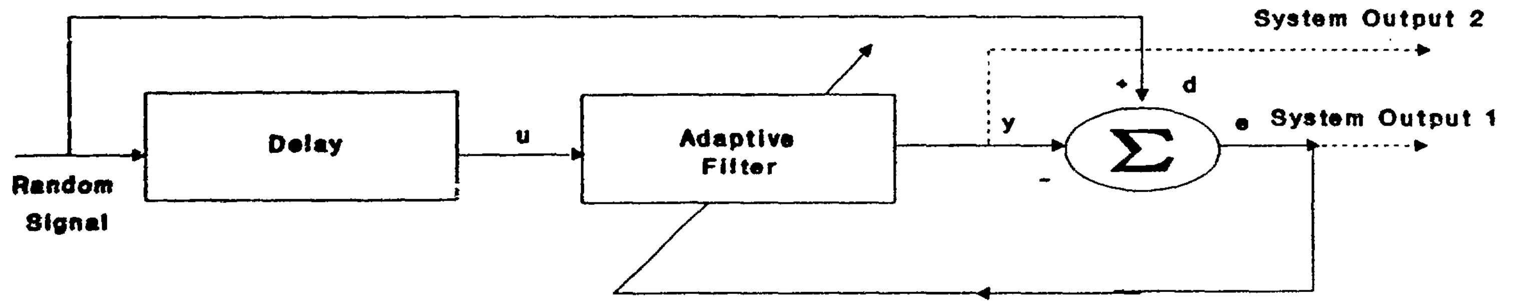


Fig. 3.4

Interference Cancelling (Class IV)

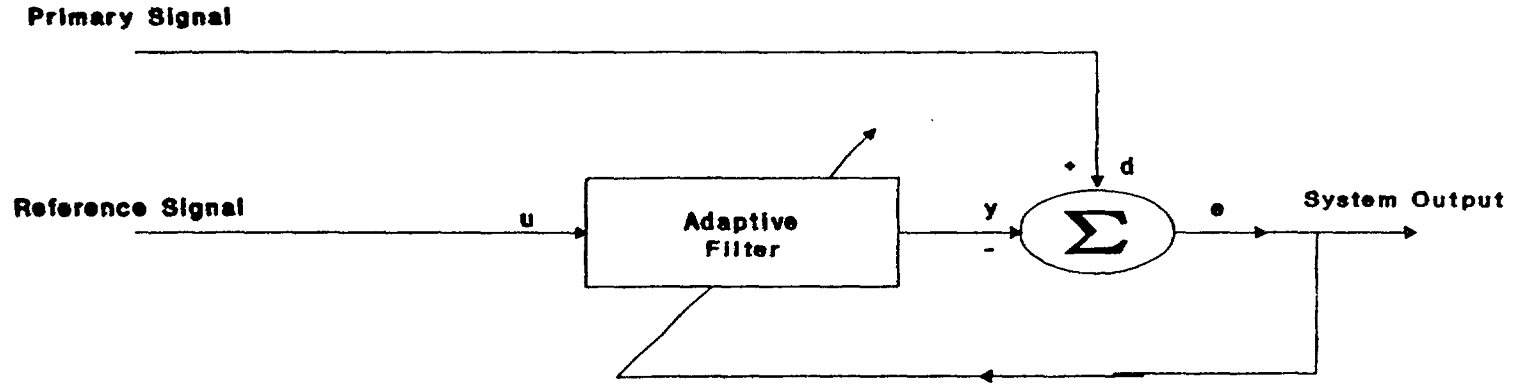


Fig. 3.5

B. ADAPTIVE NOTCH FILTERS

Notch filters for removing multiple narrow-band interference can be categorized into four broad categories illustrated in Fig 3.6 through 3.9. The first two categories, Figures 3.6 and 3.7, are cascaded second order notches with each second-order section removing one frequency. The next two categories,

Figures 3.8 and 3.9, are higher-order notches that eliminate multiple frequencies. In all of the categories, it is possible to use FIR filters (*i.e* all zero filters) which are easily pipelined and can be made truly linear phase. However, IIR notch filters require substantially fewer multipliers and adders than FIR notch filters. Thus IIR pipelining may become an important issue. In this thesis we limit our discussion only to the first two categories illustrated in Figures 3.6 and 3.7.

1. Second-Order Cascaded Notch Filters

The second-order notch filter is used in cascade and in-line with the signal as shown in Figure 3.6. The transfer function for such a notch filter is given by Kwan and Martin [Ref. 8] as:

$$\begin{aligned} H_N(z) &= 1 - H_{bp}(z) && (3.2)\\ &= 1 - \left\{ \frac{k_2}{2} \frac{(1+z^{-1})(1-z^{-1})}{1-(2-k_2-k_1^2)z^{-1}+(1-k_2)z^{-2}} \right\} && (3.3)\\ &= \frac{2-k_2}{2} \frac{1 - \frac{2(2-k_2-k_1^2)}{2-k_2} z^{-1} + z^{-2}}{1-(2-k_2-k_1^2)z^{-1}+(1-k_2)z^{-2}} && (3.4) \end{aligned}$$

For arbitrary values of k_1 and k_2, this is a symmetric notch filter with unity gain at DC and the Nyquist frequency. If k_2 is kept constant, then the 3db notch width is also kept constant. Thus k_1 may be adapted to remove one narrow-band signal. A cascade of such filters can be used to remove multiple narrow- band signals [Ref. 8]. Constants k_1 and k_2 are related to the pole radius r and the normalized pole

Cascade of Second-Order Notch Filters

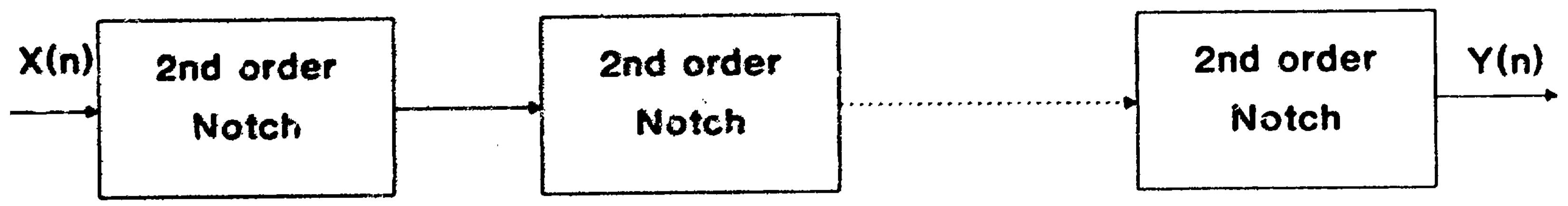


Fig 3.6

Cascade of Second-Order Cancellation Filters

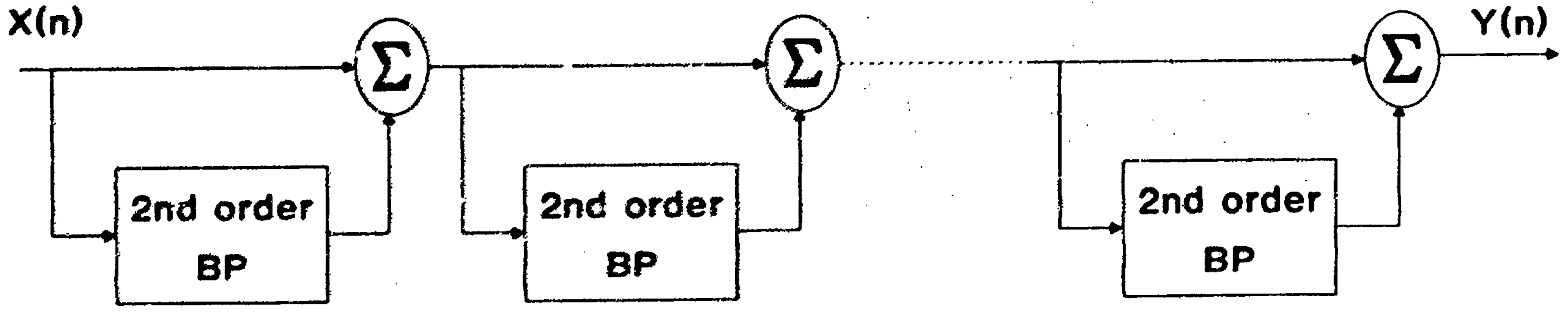


Fig 3.7

Higher-Order Notch Filter

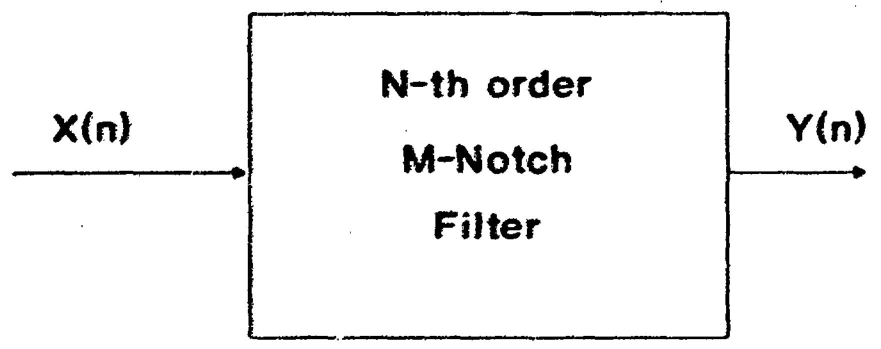


Fig 3.8

Higher-Order Signal Cancelling Filter

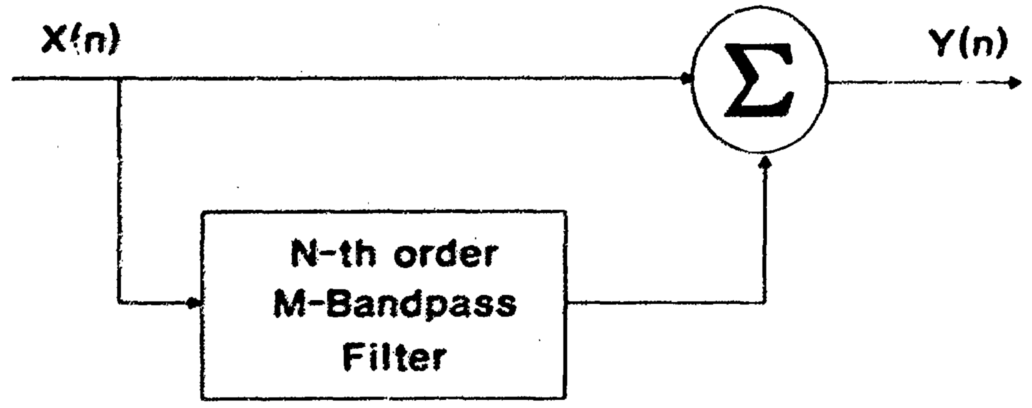


Fig 3.9

frequency θ_p as

$$k_2 = (1 - r^2) \tag{3.5}$$

$$k_1 = \sqrt{1 + r^2 - 2r\cos\theta_p} \tag{3.6}$$

It is important to bear in mind that the above transfer function has unity gain at

$$\theta_{peak} = 2\arcsin\left\{\frac{k_1}{2\sqrt{1 - \frac{k_2}{2}}}\right\} \tag{3.7}$$

which is different from θ_p [Ref. 8]

2. Second-Order Cascaded Signal Canceler

The cascaded second-order signal canceler approach shown in Figure 3.7 has the advantage that the desired signal does not pass through the adaptive filter. Instead, the band-pass filter is used to detect the narrow-band signal which is then subtracted from the desired signal. A constant 3db bandwidth notch can be achieved by selecting a band-pass filter with the transfer function:

$$H_{bp}(z) = \frac{k_2(1 - z^{-2})}{2D(z)} \tag{3.8}$$

$$= \frac{N(z)}{D(z)} \tag{3.9}$$

where $D(z)$ is the same denominator as $H_N(z)$ in equation (3.4) The signal-canceler structure is also nice for adaptation because it is relatively easy to generate sensitivity functions which are related to the gradient of $H_{bp}(z)$ with respect to the frequency parameter k_1. Figure 3.10 shows the block diagram of an adaptive version of this filter. The sensitivity function $s(n)$ is obtained by differentiating the error signal $e(n)$ with respect to the parameter k_1. This can be easily derived by noting that $e(n) = x(n) - y(n)$ and

$$\frac{\partial e(n)}{\partial k_1} = -\frac{\partial y(n)}{\partial k_1} \tag{3.10}$$

Sensitivity Function

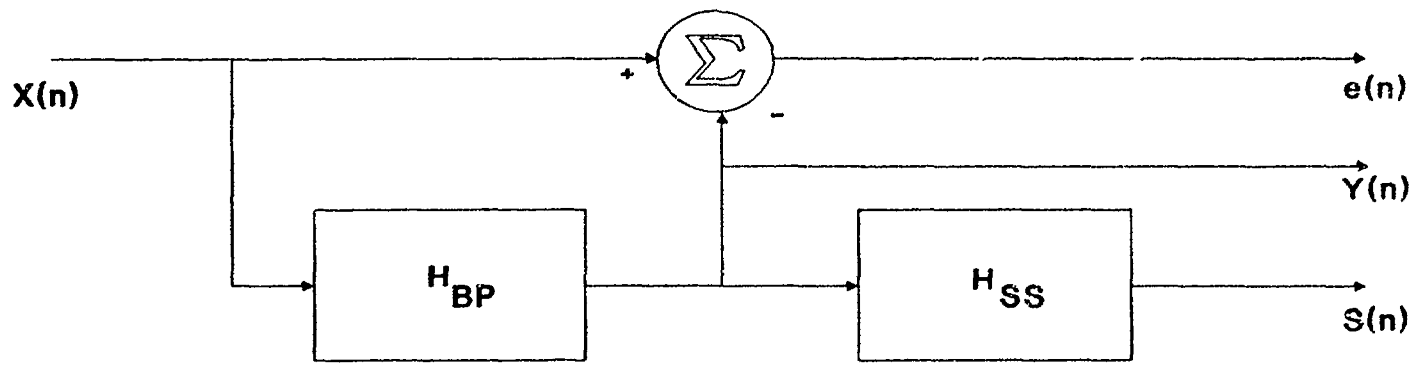


Fig 3.10

One Section of Kwan and Martin Filter

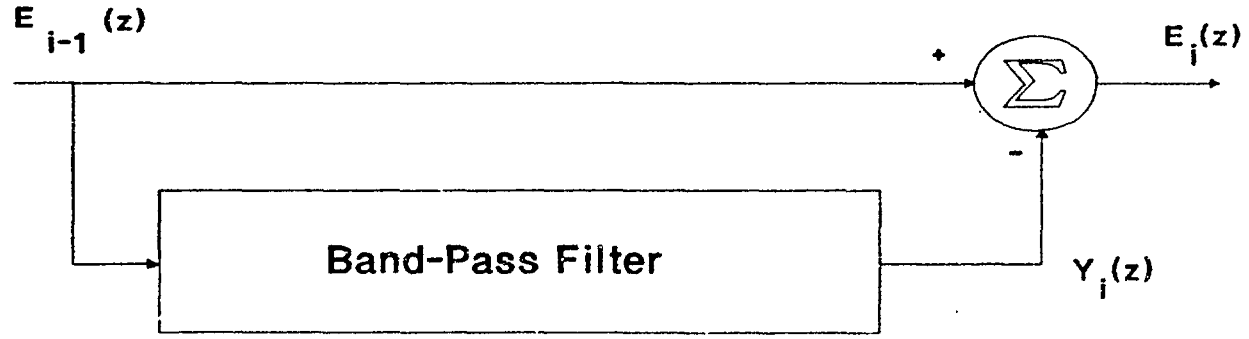


Fig 3.11

to get the above partial derivative we consider

$$\begin{aligned} y(n) &= H_{bp}(z)x(n) && (3.11) \\ &= \frac{N(z)}{D(z)}x(n) && (3.12) \end{aligned}$$

the above equation (3.12) can be easily differentiated with respect to k_1 by using the equation (2.47) to obtain:

$$\frac{\partial y(n)}{\partial k_1} = -\left[\frac{N(z)}{[D(z)]^2}2k_1 z^{-1}\right]x(n) \qquad (3.13)$$

by recognizing that $H_{bp}(z) = \frac{N(z)}{D(z)}$ from equation (3.12) the above equation can be written as

$$\frac{\partial y(n)}{\partial k_1} = -\left[H_{bp}(z)\frac{2k_1 z^{-1}}{D(z)}\right]x(n) \qquad (3.14)$$

and by defining

$$H_{ss}(z) = \frac{2k_1 z^{-1}}{D(z)} \qquad (3.15)$$

we get the sensitivity function as

$$s(n) = \frac{\partial e(n)}{\partial k_1} = H_s(z) = H_{bp}(z)H_{ss}(z) \qquad (3.16)$$

The equation (3.16) is given as a block schematic in Fig 3.10. The parameter k_1 may then be adapted by the formula:

$$k_1(n+1) = k_1(n) - \mu e(n)\frac{s(n)}{\|s(n)\|^2} \qquad (3.17)$$

a. Computing $\|s(n)\|^2$

Ideally $\|s(n)\|^2$ can be calculated by the relation

$$\|s(n)\|^2 = \sum_{i=n-N}^{n} s(i)^2 \qquad (3.18)$$

The above scheme computes the average of the past N samples. However a weighted average of the past samples with highest weight for the recent sample can be done quite easily by a first-order low pass filtering $[s(n)]^2$ as follows:

$$v_n = v_{n-1}\lambda + (1-\lambda)s^2(n) \tag{3.19}$$

However the above lowpass filtering can also be carried out by a second order filter such as:

$$v_n = \left[\frac{b_1(1-2z^{-1}+z^{-2})}{1-2r_f k z^{-1}+z^{-2}}\right][s(n)]^2 \tag{3.20}$$

$$v_n = \left[\frac{b_1(1-2\cos(5\theta)z^{-1}+z^{-2})}{1-2r_f k z^{-1}+z^{-2}}\right][s(n)]^2 \tag{3.21}$$

Yet another way to estimate $\|s(n)\|^2$ i simply:

$$v_n = \sum_{i=1}^{N}[s^i(n)]^2 \tag{3.22}$$

the above form is defined as *zero order forgetting.* It should be noted that equation (3.21) was used by Kwan and Martin [Ref. 8].

C. MULTIPLE NARROWBAND SIGNAL SUPPRESSION

The second-order implementations of section (1.) and (2.) offer considerable advantage both in hardware complexity and in adaptive performance for multiple narrow band signal suppression [Ref. 8],[Ref. 25],[Ref. 26].

1. The Kwan and Martin Filter

In a recent paper by Kwan and Martin [Ref. 8], the problem of detecting and enhancing sinusoidal signals in the presence of noise is addressed with a cascade of IIR adaptive notch filters which are used to eliminate the sinusoids. Each of the sinusoids is eliminated by a bandpass filter whose output is an enhanced version of one of the sinusoids. Hence this remarkable structure can perform both tasks with a single adaptive filter configuration which is shown to be highly robust and performs extremely well.

The major disadvantage of the Kwan and Martin structure is that the number of biquad sections needed in the adaptive filter configuration is given by N(N+3)/2, where N is the number of sinusoids to be detected and removed. This becomes impractical in real-time situations with more than 4 sinusoids due to the geometric increase in the required hardware.

a. Kwan and Martin Structure

The Kwan and Martin structure consists of a cascade of IIR notch filters one stage of which is shown in Figure 3.11. Each stage consists of a bandpass filter with zeros at DC and the Nyquist frequency and unity gain at its peak frequency ω_i. Such a filter would have the following z-domain transfer function

$$H^i_{bp} = \frac{1-r_i^2}{2}\frac{1-z^{-2}}{1-2r_i cos\theta_i z^{-1} + r_i^2 z^{-2}} \tag{3.23}$$

where

r_i = pole radius of the i-th section

$\theta_i = 2\pi\omega_i/\omega_s$

ω_i = peak frequency of the i-th section

ω_s = sampling frequency

Kwan and Martin identify two different methods for adapting the filter. Most of their derivation is based on what they call the *constant bandwidth* filter in which the pole radius r_i is a constant and only the frequency ω_i is adapted. An alternative approach which keeps a constant Q is also discussed in Kwan and Martin paper. In addition, Kwan and Martin select the adaptive quantity in such a way that it is fairly easy to determine the notch frequency from the adaptive parameter. From

Figure 3.10, we see that the notch filter for each section is the difference between 1 and the bandpass filter, hence:

$$H_N^i(z) = 1 - H_{bp}^i(z) \tag{3.24}$$

b. Calculation of the gradient

The basic structure of the Kwan and Martin adaptive filter shown in Figure 3.12 is a cascade of N sections of the form of Figure 3.7. The overall transfer function is given by

$$\begin{aligned} T(z) &= \prod_{i=1}^{N} H_N^i(z) & (3.25)\\ &= \prod_{i=1}^{N} \left(1 - H_{bp}^i\right) & (3.26) \end{aligned}$$

Kwan and Martin choose as their objective function $J(z)$ the square of the output of the final stage of the cascade:

$$\begin{aligned} J(z) &= \{E_1(z)\}^2 & (3.27)\\ &= [T(z)]^2[X(z)]^2 & (3.28) \end{aligned}$$

Hence, the gradient of the objective function $J(z)$ is given by

$$\frac{\partial J(z)}{\partial k_j} = 2E_1(z)X(z)\frac{\partial T(z)}{\partial k_j} \tag{3.29}$$

Thus in order to find the gradient with respect to the adaptive parameters k_j , we must take the partial derivative of $T(z)$ with respect to each k_j

$$\frac{\partial T(z)}{\partial k_j} = \prod_{\substack{i=1\\ i\neq j}}^{N} H_N^i(z)\frac{\partial H_N^j(z)}{\partial k_j} \tag{3.30}$$

Kwan and Martin Structure

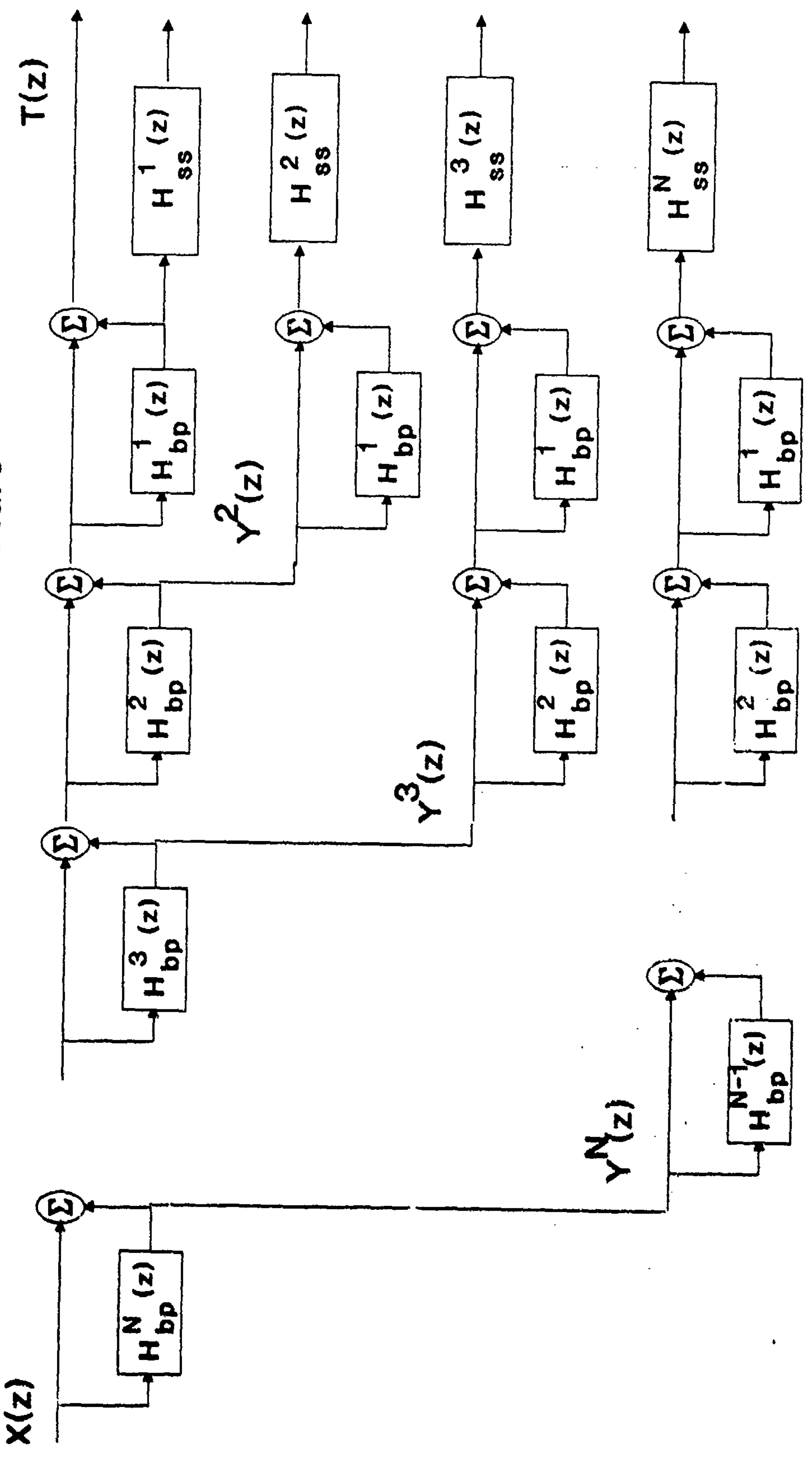


Fig 3.12

From equation (3.24) we have

$$\begin{aligned} \frac{\partial H_N^j(z)}{\partial k_j} &= \frac{\partial[1 - H_{bp}^j(z)]}{\partial k_j} && (3.31) \\ &= -\frac{\partial H_{bp}^j(z)}{\partial k_j} && (3.32) \end{aligned}$$

Using the rule for differentiation as given by the equations (2.45) and (2.47) and from equation (3.23) we have

$$\begin{aligned} \frac{\partial H_N^j(z)}{\partial k_j} &= -H_{bp}^j(z)\frac{-2k_j z^{-1}}{D(z)} && (3.33) \\ &= H_{bp}^j(z)H_{ss}^j(z) && (3.34) \end{aligned}$$

where $D(z)$ is the denominator of H_{bp}^j. H_{ss}^j is given by the equation (3.15).

Substituting equation (3.34) into equation (3.30) we obtain the over all sensitivity for the jth k_1 parameter as

$$\begin{aligned} \frac{\partial T(z)}{\partial k_j} &= \prod_{\substack{i=1 \\ i\neq j}}^{N} H_N^i(z)H_{bp}^j(z)H_{ss}^j(z) && (3.35) \\ &= \left[\prod_{i=1}^{j-2} H_N^i(z)\right]\left[\prod_{i=j-1}^{N} H_N^i(z)\right] H_{bp}^j(z)H_{ss}^j(z) && (3.36) \end{aligned}$$

By recognizing that

$$Y^j(z) = \left[\prod_{i=j-1}^{N} H_N^i(z)H_{bp}^j\right] \qquad (3.37)$$

and by substituting equation (3.37) in the equation (3.36) we get

$$\frac{\partial T(z)}{\partial k_j} = \left[\prod_{i=1}^{j-2} H_N^i(z)\right] H_{ss}^j(z) \qquad (3.38)$$

Figure 3.12 shows the Kwan and Martin realization of the complete adaptive system. The difficulty in the Kwan and Martin approach is in the generating of the product of notch filters without the notch filter j, as required in equation (3.36). To generate this product for each section, would require $N - 1$ biquads per section resulting in a total of $N(N-1)$ biquads just to generate the product. Kwan and Martin are able to reduce this by using the output of the bandpass filter as the input to their cascade via equation (3.37). Since this output already has $(j-1)$ of the required $H_N^i(z)$ factors in it, only $(N-j)$ additional biquads are needed for a total of $0.5(N^2-N)$. Adding this to the N biquads required to realize the cascade of notch filters and the N biquads required to realize the $H_s^j(z)$ factors, yields a total number of biquads given as $0.5(N+3)N$.

2. The New Structure

Figure 3.13 shows the improved adaptive notch filter structure [Ref. 25],[Ref. 26]. The key to the improvement is the recognition that the output $E_1(z) = T(z)X(z)$ for the cascade of the notch filters can be written both as a product of the individual notch filter section transfer functions $H_N^i(z)$ times the input and in terms of the input $X(z)$ minus the outputs $Y^i(z)$ of the bandpass filters

$$\begin{aligned} E_1(z) &= T(z)X(z) && (3.39)\\ &= \left[\prod_{i=1}^{N} H_N^i(z)\right] X(z) && (3.40)\\ &= X(z) - \left(\sum_{i=1}^{N} Y^i(z)\right) && (3.41) \end{aligned}$$

To get the product of $H_N^i(z)$ without the term $i = j$ as required in the equation (3.36), we may use equation (3.41) to simply add back the term $Y^j(z)$

$$\begin{aligned} \left[\prod_{\substack{i=1\\ i\neq j}}^{N} H_N^i(z)\right] X(z) &= X(z) - \left[\sum_{\substack{i=1\\ i\neq j}}^{N} Y^i(z)\right] && (3.42)\\ &= E_1(z) + Y^j(z) && (3.43) \end{aligned}$$

Figure 3.13 makes use of this fact to generate the gradient needed for the

New Structure

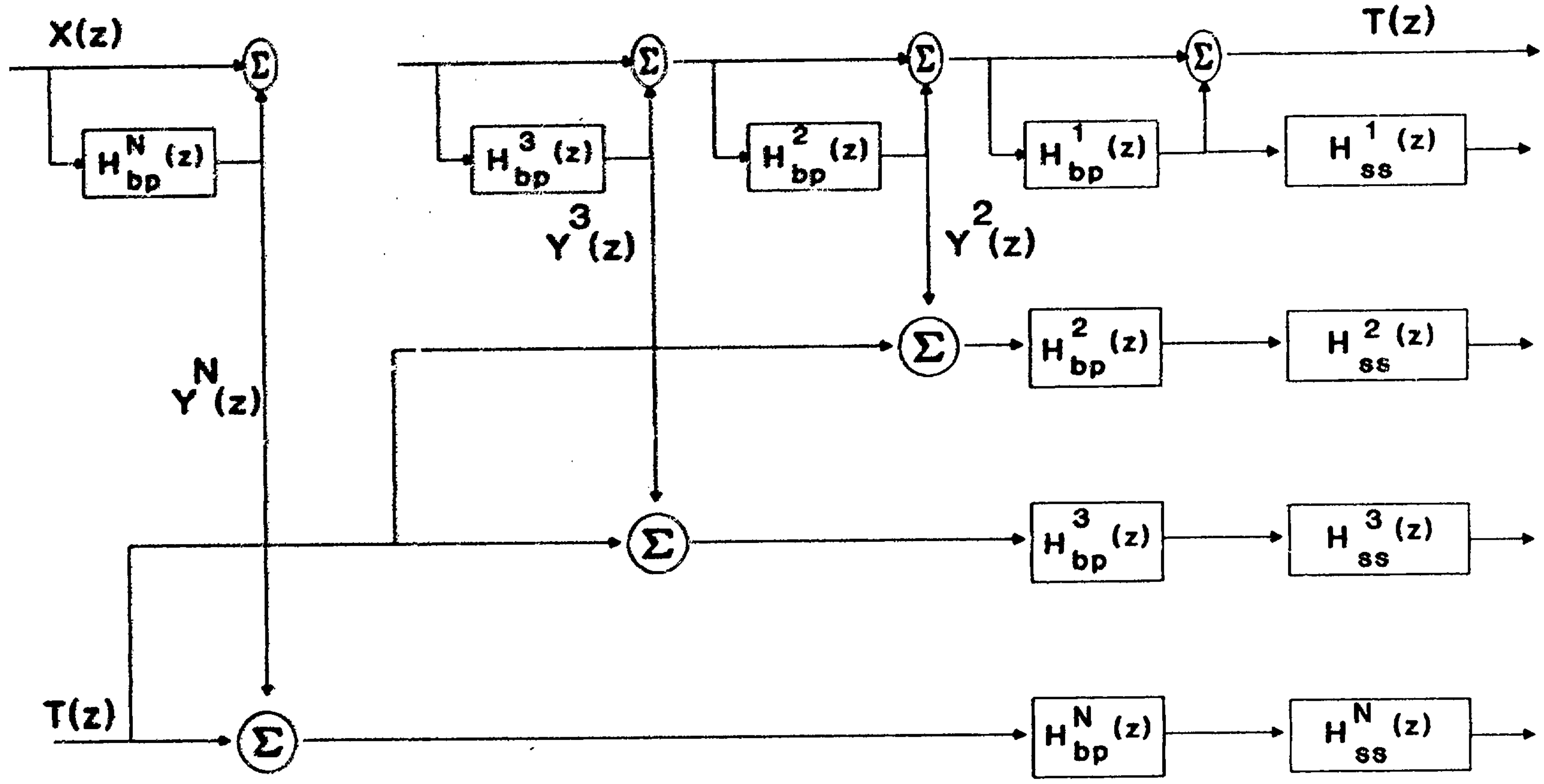


Fig 3.13

adaptive process. From the Figure 3.13, we can see that the total number of biquads required is N for the cascade of notch filters plus 2N for the $H_{bp}^{i}(z)H_{s}^{i}(z)$ required for adaptation, minus 1 at the last stage, since the last stage does not need the extra $H_{bp}^{i}(z)$). Thus we have the number of biquads required in the new structure is (3N-1).

IV. MODELLING and SIMULATION

A. MODELLING OF SIGNALS

In order to test various algorithms and to evaluate their probable performance in the real environment, it was necessary to develop a meaningful simulation of the real situation. To achieve this, the following four testing categories were developed:

(i) Sinusoidal signals with white gaussian noise

(ii) Narrow Band Noise with white gaussian noise

(iii) Bi-Phase Shift Keying (BPSK) sequence

(iv) Frequency Shift Keying (FSK) sequence

1. Sinusoidal Signals

In order to generate sinusoidal signals with minimum computational burden placed at different frequencies $\frac{\theta}{360}f_s$, a second order AR process

$$x_k^i = 2cos(\theta_i)x_{k-1}^i - x_{k-2}^i \tag{4.1}$$

was used. Initial conditions are very important and they are chosen such that $x_{-1} = 0$ and $x_{-2} = -sin(\theta_i)$ giving a unit amplitude sinusoidal signal. The θ_i value is between 0 to 180 and n is the number of frequencies desired. The required signal y_k needed to input into the adaptive algorithm is given as

$$y_k = \sum_{i=1}^{n} x_k^i + \gamma_k \tag{4.2}$$

where γ_k is a white gaussian noise $N(0, \sigma^2)$.

2. Narrow Band Noise

The narrow band noise signal is generated using the difference equation

$$x_k^i = 2rcos(\theta_i)x_{k-1}^i - r^2 x_{k-2}^i + u_k^i \tag{4.3}$$

where θ_i decides the placement of the noise in the spectrum and r controls the bandwidth of the noise. The u_k^i is simply a uniformly distributed noise taken at different instants. The desired signal y_k is obtained via

$$y_k = \sum_{i=1}^{i=n} z_k^i + \gamma_k \tag{4.4}$$

3. Bi-Phase Shift Keying Sequence

The generation of BPSK signal has three distinct three parts:

(i) Generation of Random Binary Sequence(RBS) is achieved by passing uniformly distributed noise through a hardlimiter (An important note is that the interval between the two consecutive bits of RBS is $\frac{1}{f_b}$);

(ii) Another sequence of binary numbers is a *spreading code or sequence*. The specific sequence used in a given communications system is normally not available to anyone but to the designated receiver. (In this particular simulation we have generated the spreading sequence by passing uniformly distributed noise through a hardlimiter. The bit interval is $\frac{1}{f_c}$);

(iii) Phase encoding (*i.e.*) mapping the given binary signal which is the *exclusive or* of (i) and (ii) as 0 or π at *appropriate* sample time.

The output of the first hardlimiter is stored in an array x. Output of the second hardlimiter is stored in the array y This information is retrieved by a subtle use of the array index given as $i = kf_b$ where i is the index, k is the discrete sample number and f_b is the bit rate of the intelligence. Similarly another index j is generated using $j = kf_c$ where f_c is the chip frequency. Generation of the indices is the *key thing in this simulation*. The desired signal now is given by the equations

$$w_k^p = Acos(2\pi f_0 k + \phi_k) \tag{4.5}$$

$$\phi_k = [x(i) \oplus y(j)]\pi \tag{4.6}$$

$$\alpha_k = \sum_{p=1}^{p=n} w_k^p \tag{4.7}$$

where i and j are indices of the arrays as defined earlier. It is an important point to note that the characteristic of the signal w_k^p are defined by the parameter $p = \{f_0, f_c, f_b\}$.

This signal is not really a simple BPSK signal but it has an additional feature of spreading the spectrum by controlling the chip-frequency and carrier frequency and information rate. A block diagram of the scheme is given in Fig 4.1.

The desired signal y_k is given by:

$$y_k = \alpha_k + \beta_k \tag{4.8}$$

where α_k is the set of narrowband BPSK signals placed at different places in the frequency spectrum and β_k is the broad band BPSK signal generated for a specific p value.

4. Frequency Shift Keying Sequence

Generating this sequence needs a random binary intelligence signal. This was once again is achieved by passing a uniformly distributed noise through a hardlimiter. The output of this hard limiter stored in an array x. An index i is chosen such that $i = kf_b$ where f_b is the baud-rate of the information and k is discrete sample number. Now the desired signal is generated via

$$s_k = 2cos(\theta_k)s_{k-1} - s_{k-2} \tag{4.9}$$

$$\theta_k = \theta + \delta x(i) \tag{4.10}$$

$$y_k = s_k + \gamma_k \tag{4.11}$$

where θ is the carrier frequency and δ is the depth of the frequency modulation. Initial conditions are very important and they are chosen such that $s_{-1} = 0$ and $s_{-2} = -sin(\theta)$ giving an unit amplitude sinusoidal signal.

DSSS BPSK Generator

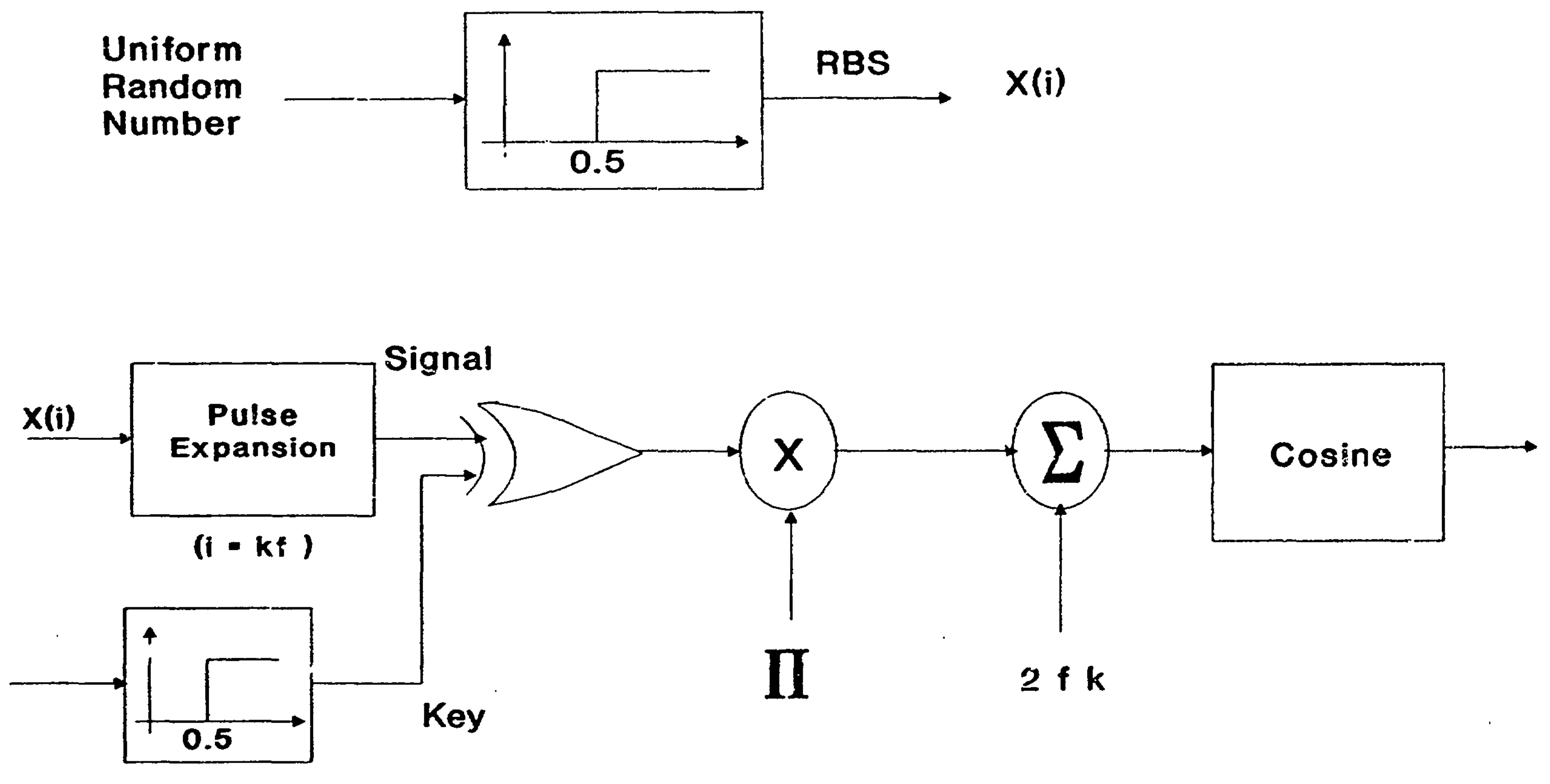


Fig. 4.1

B. SIMULATION

The adaptive digital filter algorithm as described in the earlier chapters is simulated and tested using synthetic data. Simulation was carried out on a VAX 11/785 computer using Fortran 77. Listing of the program used is enclosed in the appendix. The adaptive filter parameters of interest are

(a) Sharpness of the notch filter defined by pole position ($r^2 = 1 - \kappa_2$)

(b) Step size in the parameter update procedure (μ)

(c) Time constant of the fading filter (λ) (refer to equation 3.19)

(d) Model order (N= # of 2nd order filters)

(e) Order of the incoming signal or number of interferers (m)

1. Kwan & Martin Algorithm

In simulating this algorithm, the most important thing is the implementation of the notch filter. Fig 4.2 gives the structure of the algorithm. Let $x(n)$ be the input to the Adaptive Filter and $e(n)$ the desired output. This desired output is obtained by passing the input $x(n)$ via a cascade of N notch filters as

$$e(n) = \left\{H_N^N(z)H_N^{N-1}\cdots H_N^1(z)\right\}x(n) \tag{4.12}$$

Output at the intermediate jth section of the band-pass filter is designated as $y^j(n)$ as shown in Fig 4.2 Then the sensitivity of the κ_1^j parameter, is obtained by passing this $y^j(n)$ through another cascade of $(j-1)$ notch filters given as

$$s^j(n) = \left\{H_N^{j-1}H_N^{j-2}\cdots H_N^1 H_{s s}^j\right\}y^j(n) \tag{4.13}$$

The filter transfer functions $H_N^j(z)$ and H_{ss}^j were defined in the earlier chapter by equations (2.21) and (2.15).

Kwan and Martin Structure

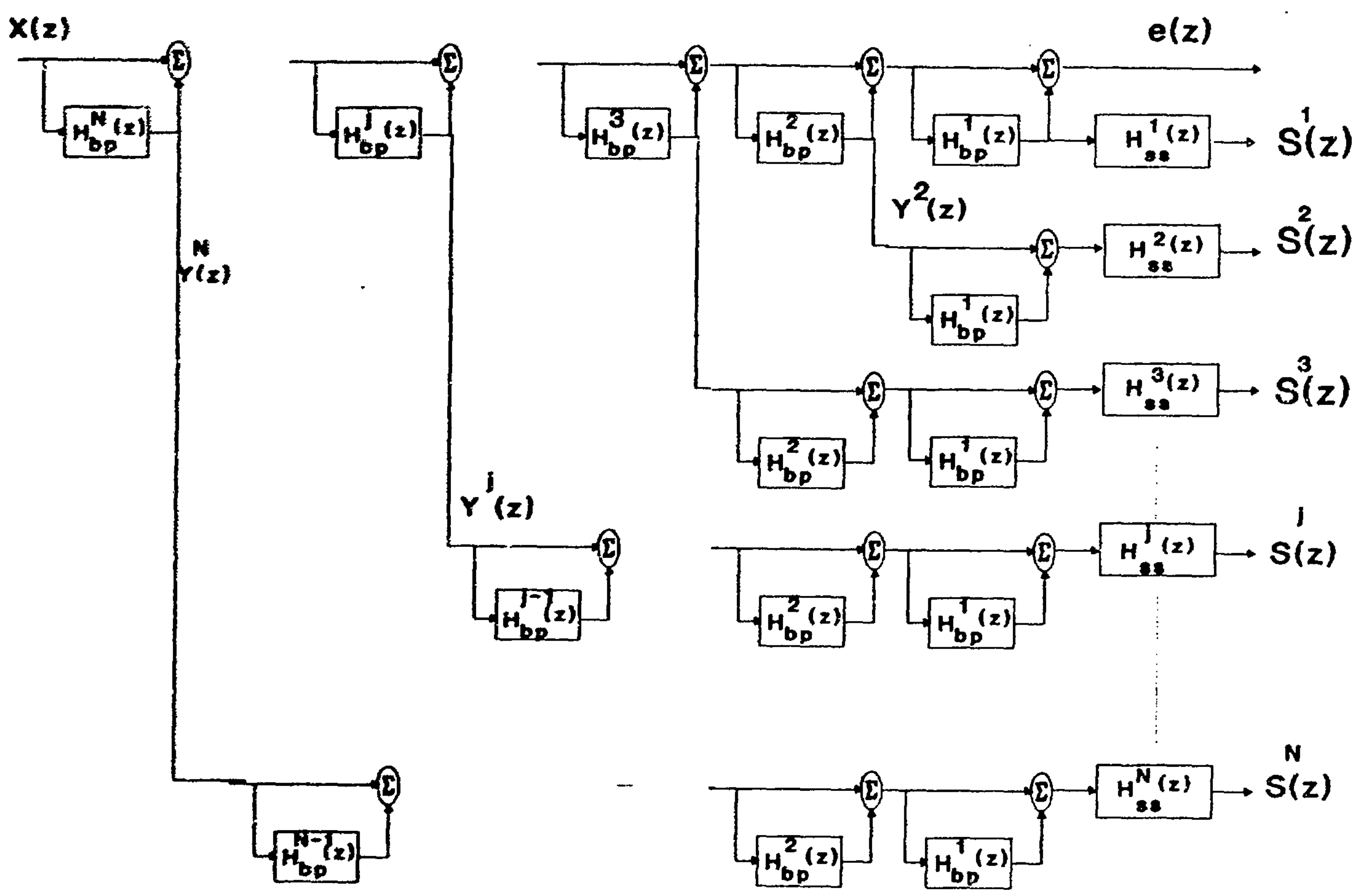


Fig 4.2

2. The New Algorithm

The primary difference between the New Algorithm and the Kwan and Martin algorithm is in the method of obtaining the sensitivity of the jth coefficient. This difference can be seen in the Fig 4.3. The output $y^j(n)$ at the jth notch Fig 4.3. filter is added to the over-all output $e(n)$ and this summed output is passed through a cascade of *only two* filters as shown in Fig 4.3 and can be given as

$$s^j(n) = \left[H^j_{bp} H^j_{ss}\right]\{e(n) + y^j(n)\} \tag{4.14}$$

3. Forgetting Filters

For parameter incrementation via equation (3.17) we need to obtain $\|s^2(n)\|$. This could be achieved by

a) Zero order forgetting using equation (3.22)
b) 1st order forgetting using equation (3.19)
c) 2nd order forgetting using equation (3.21)

4. Stability

The parameter incrementation given by the equation (3.17) can be recast by using either of the forgetting filters given above as

$$\kappa^i_1 = \kappa^i_1 - \mu[v^i]^{-1} s^i(n)e(n) \tag{4.15}$$

It is very important to note that the above equation (4.15) has a close resemblance with the ML Estimator. In the MLE case the stability while incrementing the parameter is an important factor. A similar problem exists in the current incrementation procedure but the problem is solved by a suitable choice of parameters and modifying the incrementation procedure by adding an additional factor to equation (4.15) as follows:

$$\kappa^i_1 = \kappa^i_1 - \mu e(n)s^i(n)\left[\frac{1 + \frac{v^i}{v_{max}}}{v_{min} + v^i}\right] \tag{4.16}$$

New Structure

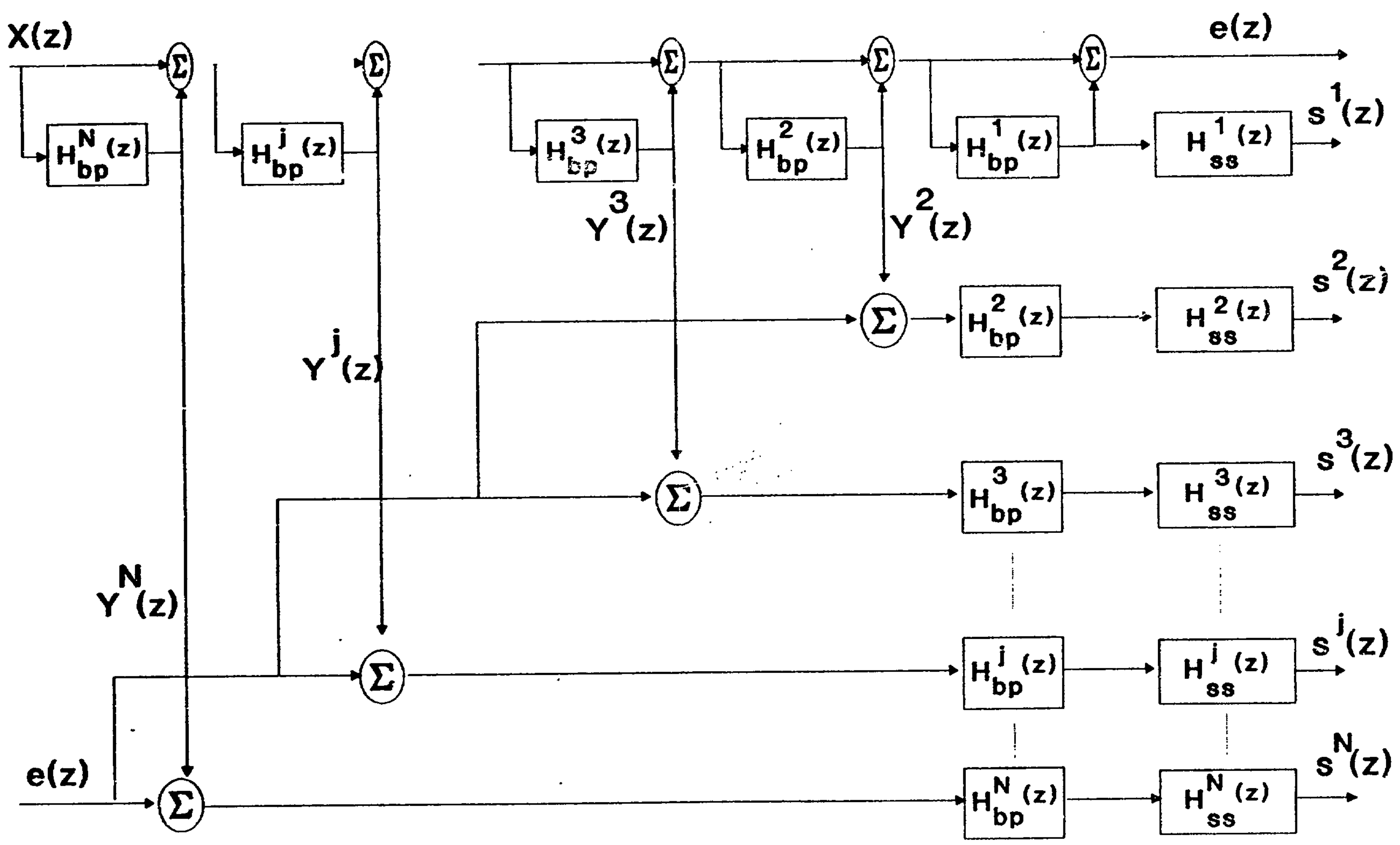


Fig 4.3

This modification also protects against possible overflow or underflow while computing $[v^i]^{-1}$. In addition simulations indicate that in the case of zero-order forgetting, this factor is required in order to obtain convergence. When v_{min} and v_{max} are zero and infinity respectively, equation (4.16) reduces to equation (4.15).

In this incrementation procedure two forms of the denominator polynomial of the $H_{bp}(z)$ were considered:

$$D^i(z) = 1 - (2 - k_2 - k_1^2)z^{-1} + (1 - k_2)z^{-2} \tag{4.17}$$

and

$$D^i(z) = 1 - 2\kappa_1^i r z^{-1} + r^2 z^{-2} \tag{4.18}$$

where $\kappa_1^i = cos(\theta^i)$.

While using the incrementation procedure, under transient conditions poles of the $D^i(z)$ cross the unit circle causing instability problems. This condition was averted by checking the pole position after incrementation and if unstable then incrementation is modified. Checking this condition for $D^i(z)$ given by equation (4.17) calls for solving a quadratic equation at every parameter increment and examining the pole position. However this check is very simple for $D^i(z)$ given in equation (4.18), since we need only to check κ_1^i by maintaining $|\kappa_1^i| \leq 1$. Furthermore the value of r must be positive and less than unity for stability. The choice of the r is very important for proper fast convergence. Fig 4.4. shows the frequency response of band-pass filter for different values of r The 3dB band-width of the band pass filter is related to r and is given [Ref. 29] by

$$Band - width = cos^{-1}\left\{\frac{2r^2}{1 + r^4}\right\} \tag{4.19}$$

Note that under limiting conditions, the band-width is essentially zero. In this simulation it was assumed a value of r as 0.95.

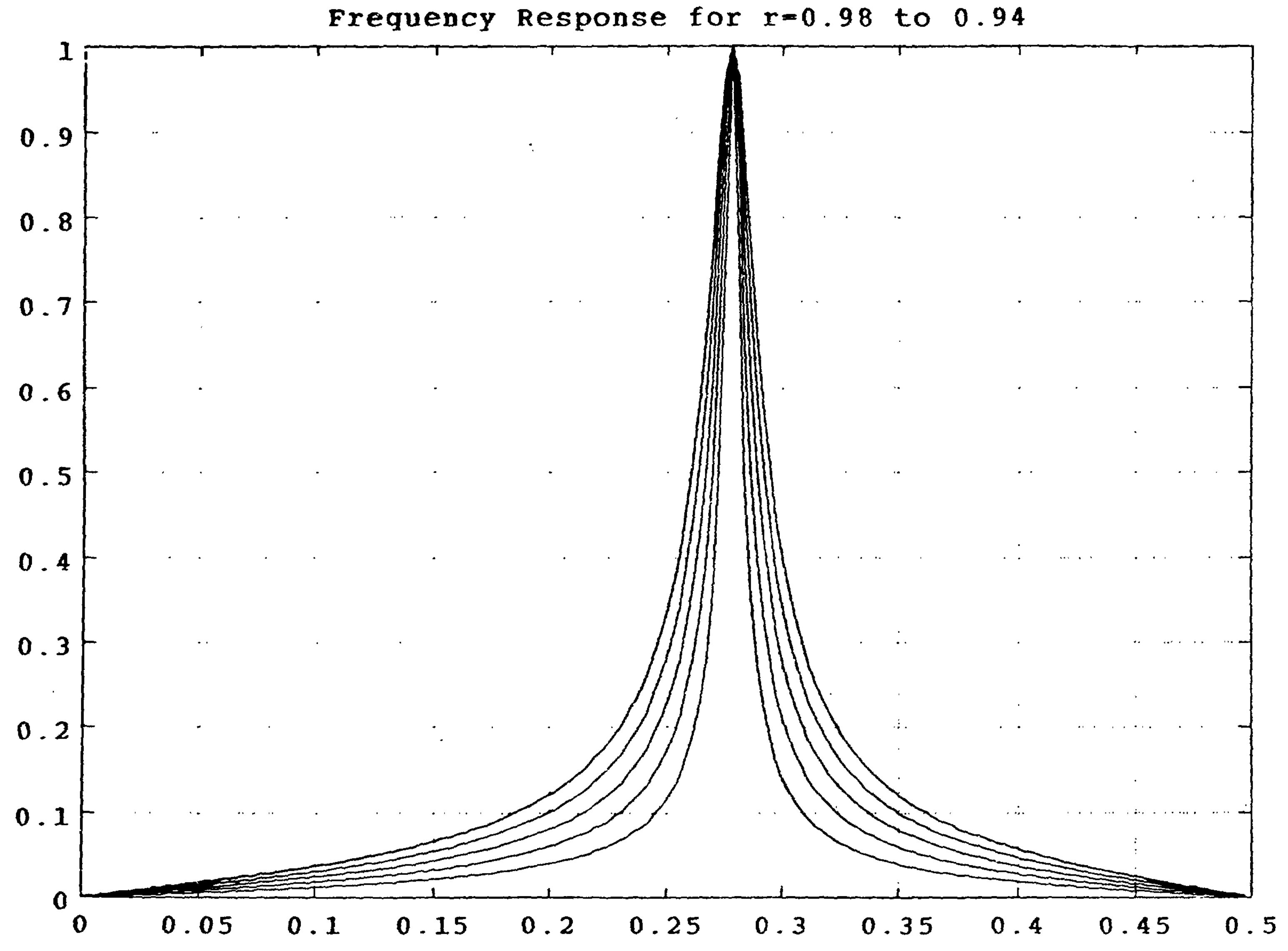

Frequency Response for r=0.98 to 0.94
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
0.4
0.45
0.5

Fig 4.4

A large number of simulations [Ref. 29] were carried out using sinusoidal inputs with and without noise for different m and N values. In these simulations the denominator polynomial used was given by equation (4.18). The results were tabulated in Table 3.1. From Table 3.1 it is seen that the choice of the step size is an important factor and step-size must be optimized for a specific number of sections N. The values of v_{min} and v_{max} did not pose any problem while incrementing using 1st order or 2nd order forgetting filters. It seems that a 1st order forgetting filter is more effective than either zero-order or a 2nd order filter.

After fixing the values of μ and λ the algorithm was tested using Narrow Band Noise signal y_k for its performance. This signal was used only to tune the value of r (*i.e.* sharpness of the notch filter). The following simulation results are based on parameter adaptation for the $D^i(z)$ given by equation (4.17).

5. Response for Sinusoidal signal

A sample simulation output due to sinusoidal signal was shown in Fig [4.5 - 4.7]. The input signal is composed of 3 sinusoids with normalized frequencies $\frac{30}{360}f_s$, $\frac{80}{360}f_s$, and $\frac{120}{360}f_s$ and WGN with $\sigma = 1$. The spectrum of this signal is shown in Fig 4.5. After passing this signal through the adaptive filter we could see that interferers were removed and only the noise was left behind. This is clearly shown in Fig 4.6. The adaptation process is shown in Fig 4.7.

6. Response for Narrow Band Noise

In this testing category sinusoids were replaced by narrow-band signals. These signals were generated via equation (4.4). Superimposed on this signal, WGN with $\sigma = 1$ was added. A typical spectrum of this signal is shown in Fig 4.8. This signal was passed through the adaptive filter and these narrow band interferers are notched out and only WGN is left out as shown in the Fig 4.9. Fig 4.10 shows the corresponding parameter convergence.

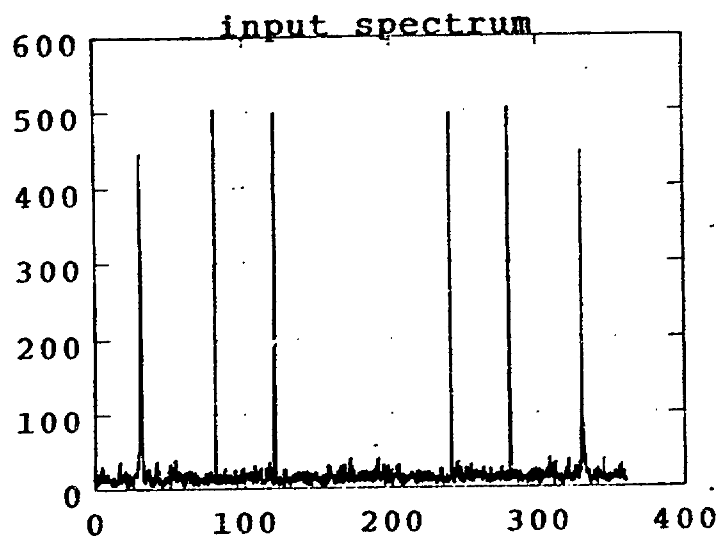


Fig 4.5

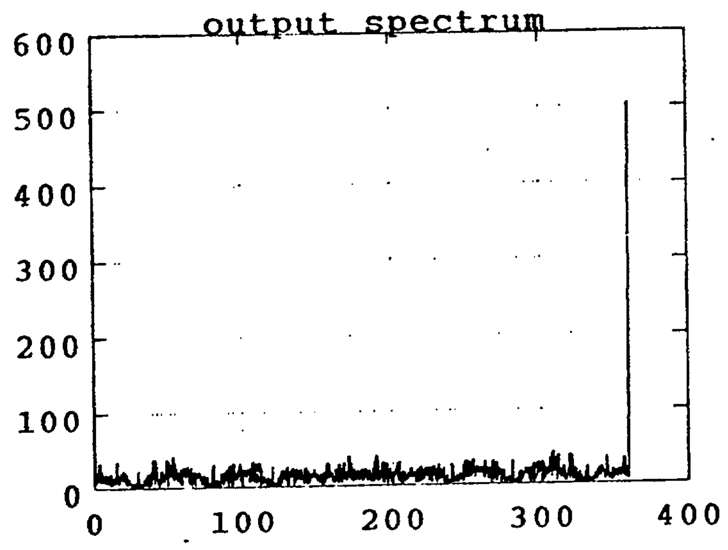


Fig 4.6

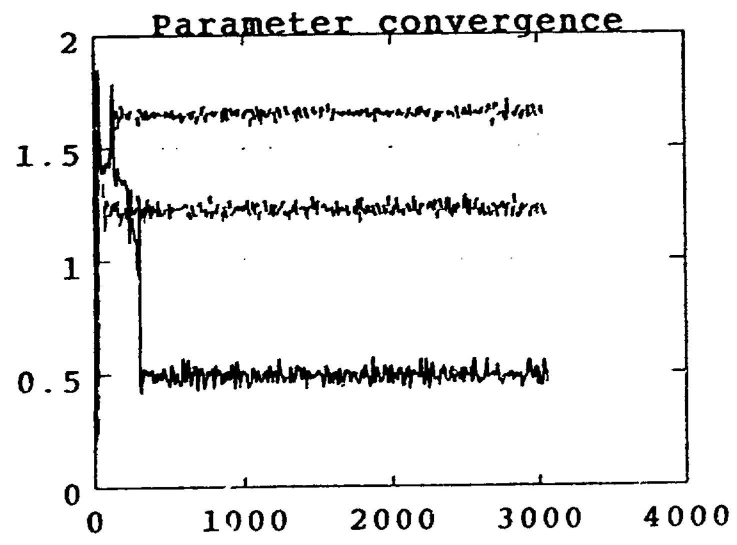


Fig 4.7

```
Input .....Sinusoidal signals
Number of interferers ...3
Samples per carrier cycle of the
interferers ........12.0,4.5,3.0
Variance of the noise .....1.0
Adaptive filter step ....0.02
Adaptive filter r .......0.9
```

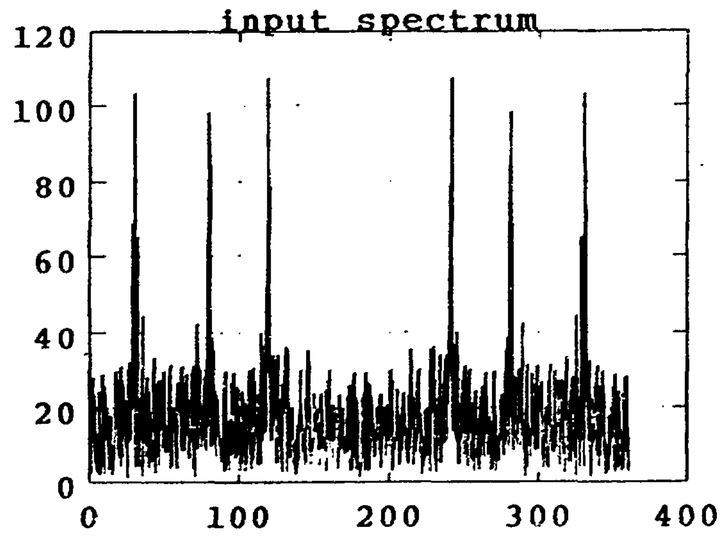


Fig 4.8

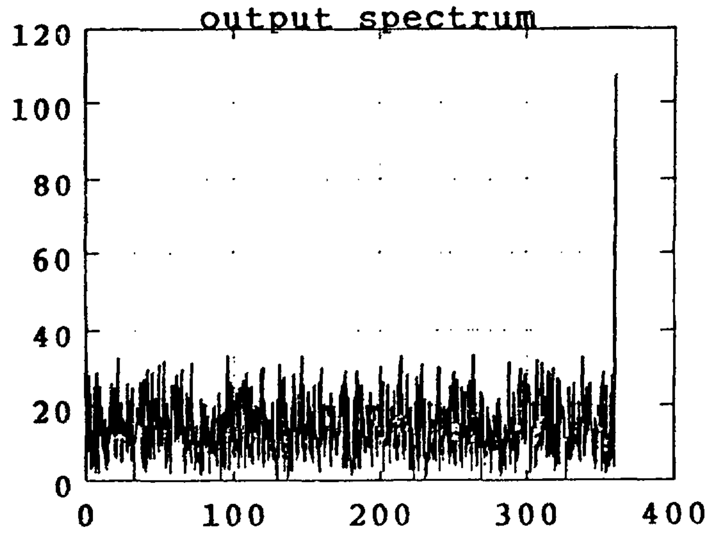


Fig 4.9

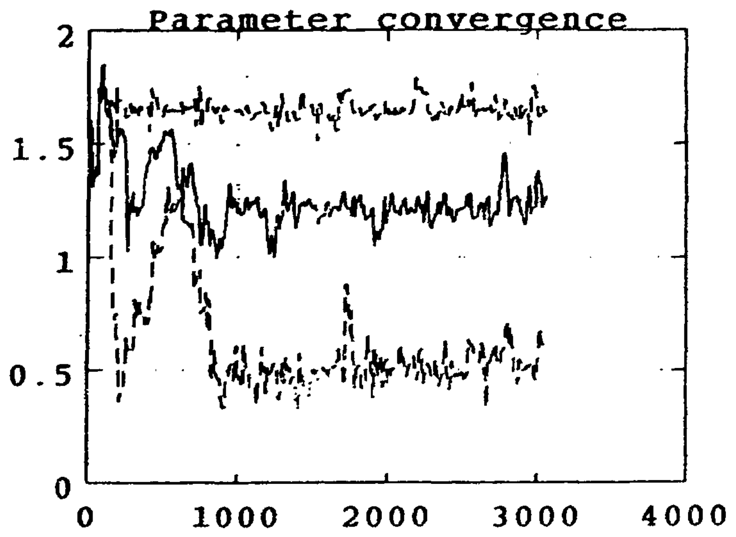


Fig 4.10

```
Input ......Narrow Band Noise
Number of interferers ...3
Samples per carrier cycle of the
interferers .......12.0,4.5,3.0
Pole radius of the Narrow Band
interferers ............0.95
Variance of the noise .....1.0
Adaptive filter step ....0.01
Adaptive filter r .......0.9
```

7. Tracking FSK Signal

Transient behavior of the adaptive filter was studied by applying an FSK signal (generated via equation 4.11). In the case of an FSK signal, signal spectrum constantly changes. This property is useful for testing the tracking ability of the adaptive filter. In fact a WGN with $\sigma = 1$ was also added to the signal. An adaptive filter then was used to demodulate the signal, by tracking the spectrum. This is shown in Fig 4.13.

8. Suppressing BPSK Interference

Having seen the performance of the adaptive filter under various conditions, it is only needed to test under an additional constraint of broad band noise. In this case, the broad-band signal was swamped by narrow-band interferers and WGN. A typical broad-band signal is generated by using a BPSK signal generator via equation (4.8). In this we have used carrier at 0.25 cycle/sample and the chip frequency at 0.125 cycles/sample. The narrow-band interference signals are also generated by using the same BPSK signal generator, but with different parameters. Interferers are chosen such that the chip frequency is fixed at $\frac{1}{200}f_s$ cycles/sample, while carriers were chosen at $\frac{30}{360}f_s$, $\frac{80}{360}f_s$, and $\frac{120}{360}f_s$ cycles/sample. To these interferers and broad-band signal, a WGN with $\sigma = 1$ was added. The composite signal spectrum is shown in Fig 4.14, indicating the three interferers, broad-band signal, and WGN. This signal was passed through the adaptive filter. Output of the filter is shown in Fig 4.15. In this figure it is clear that the interferers were removed by the filter and only the desired signal was left behind.

9. Model order mismatch

The model order of the algorithm was fixed at 3 while BPSK signals were generated with 2 interferers *i.e.* $m = 2$ and $N = 3$. Fig 3.16 shows the parameter con-

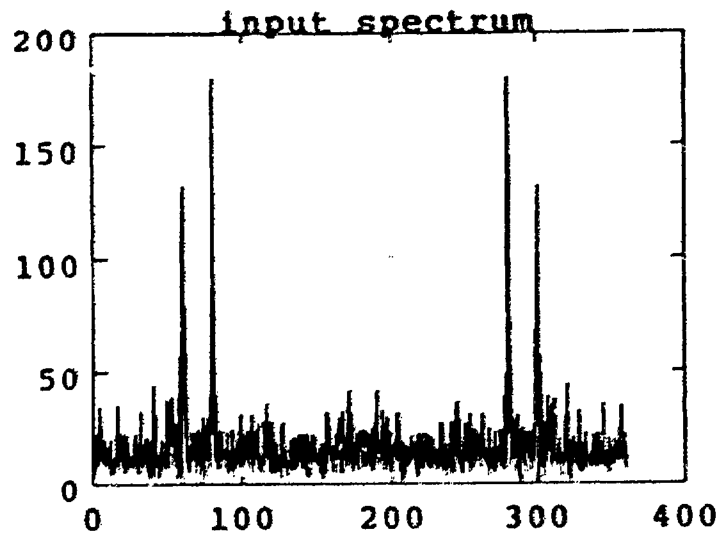


Fig 4.11

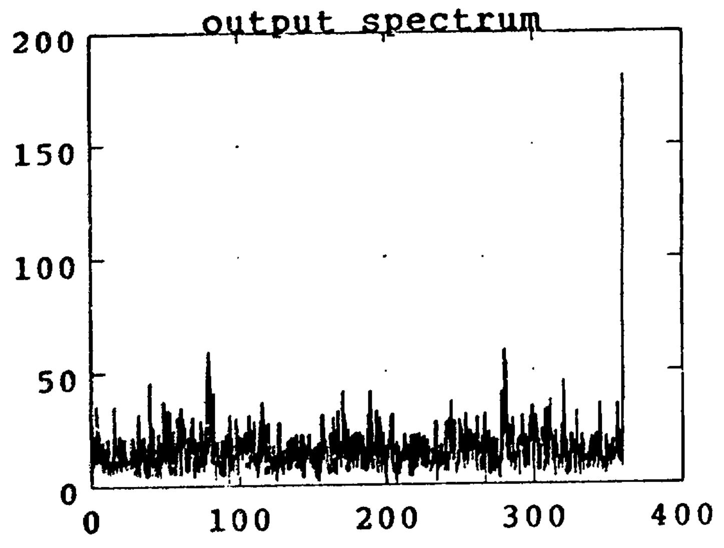


Fig 4.12

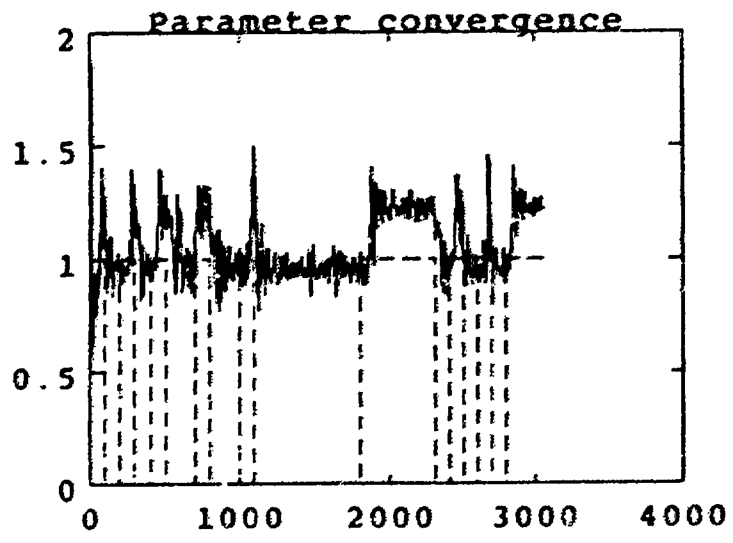


Fig 4.13

```
Input .....FSK signal
Samples per carrier cycle of the
mark and space ........6.0,4.5
Variance of the noise .....1.0
Adaptive filter step ....0.02
Adaptive filter r .......0.9
```

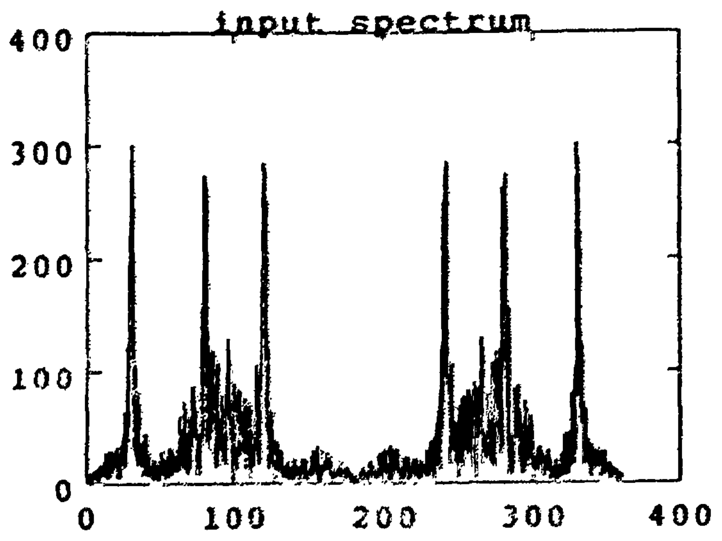


Fig 4.14

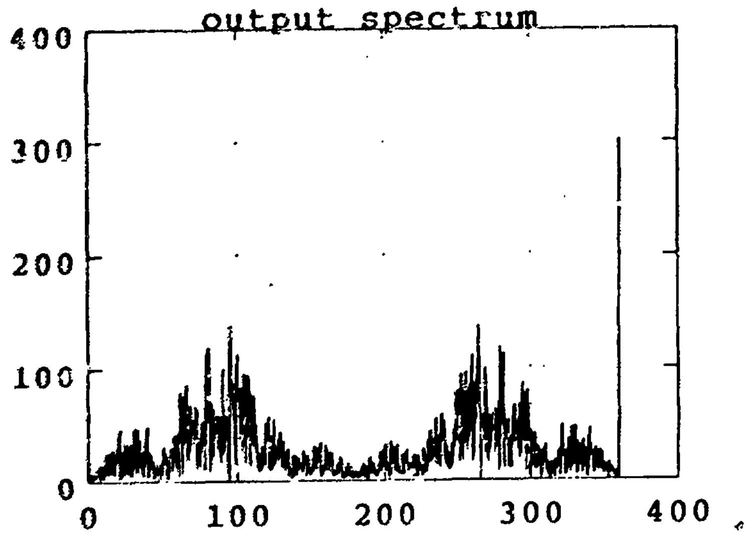


Fig 4.15

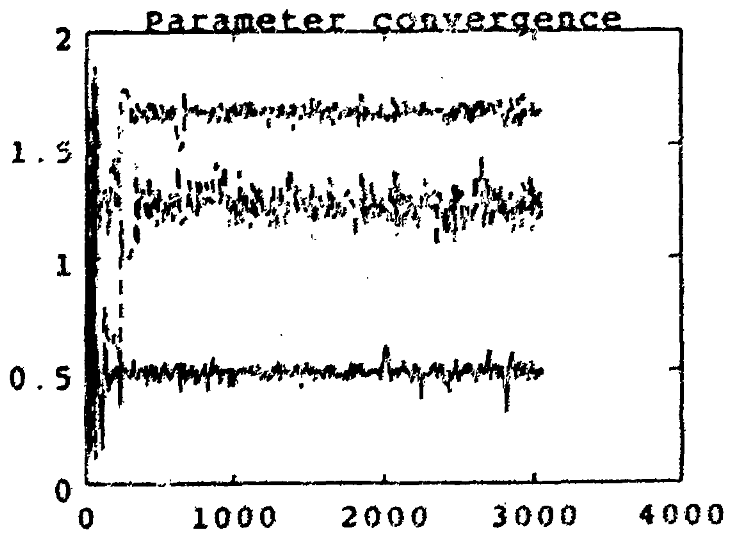


Fig 4.16

```
Input ..................DS-BPSK
Number of interferers ...3
Samples per carrier cycle of the
interferes .............12.0,4.5,3.0
Samples per Chip bit of
interferers ............200,200,200
Broad band signal samples per
carrier cycle .........  4.0
Broad band signal samples per
chip bit ...............8.0
Adaptive filter step ....0.02
Adaptive filter r .......0.9
```

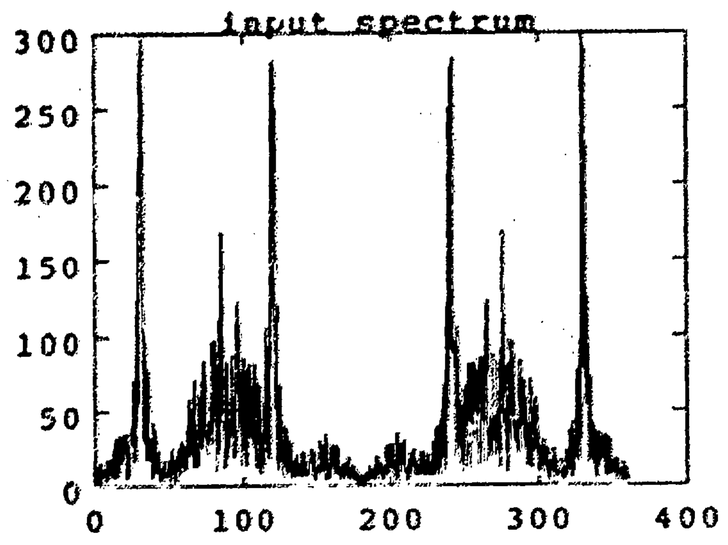


Fig 4.17

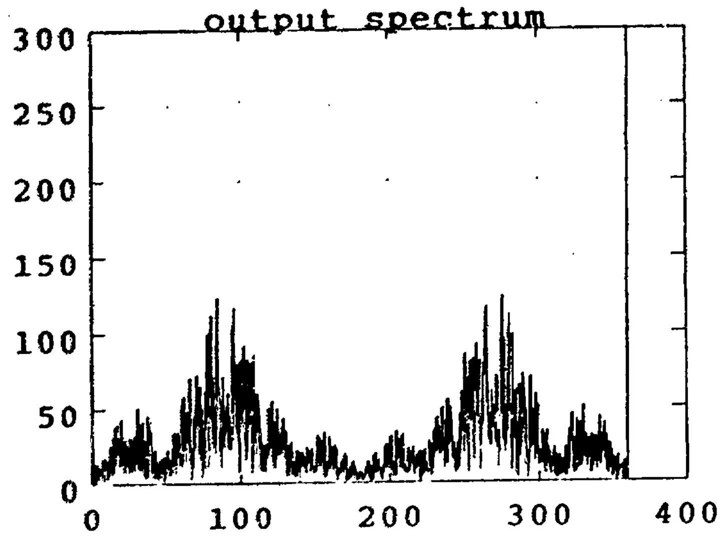


Fig 4.18

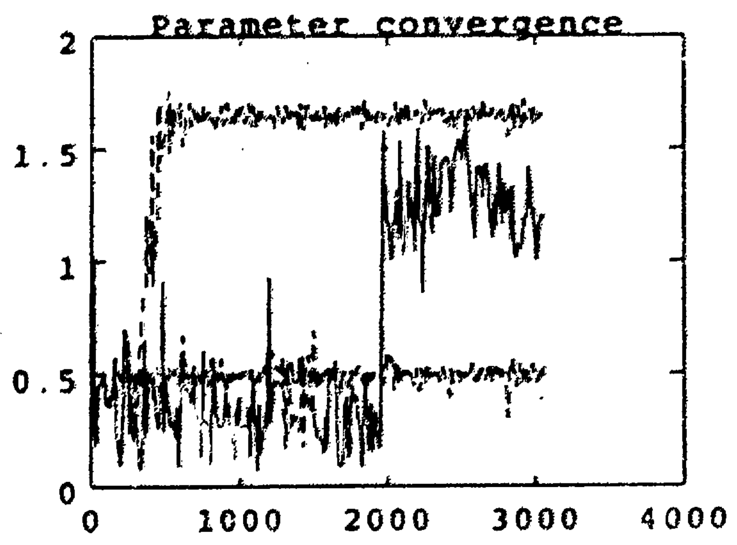


Fig 4.19

Model mismatch m=2 and n=3	
Input	DS-BPSK
Number of interferers	2
Samples per carrier cycle of the interferes	12.0,3.0
Samples per Chip bit of interferers	200,200,200
Broad band signal samples per carrier cycle	4.0
Broad band signal samples per chip bit	8.0
Adaptive filter step	0.02
Adaptive filter r	0.9

vergence under these conditions and Fig 3.19 shows for $m = 1$ and $N = 3$ conditions. We could notice that free extra parameter(s) wandering due to the mismatch. This has the effect of notching out the desired signal. This can be solved by deliberately introducing a known BPSK signal at fixed place.

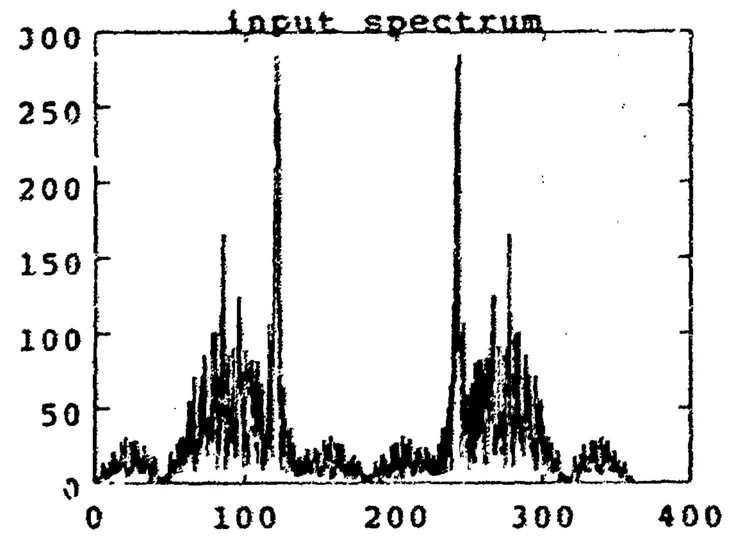


Fig 4.20

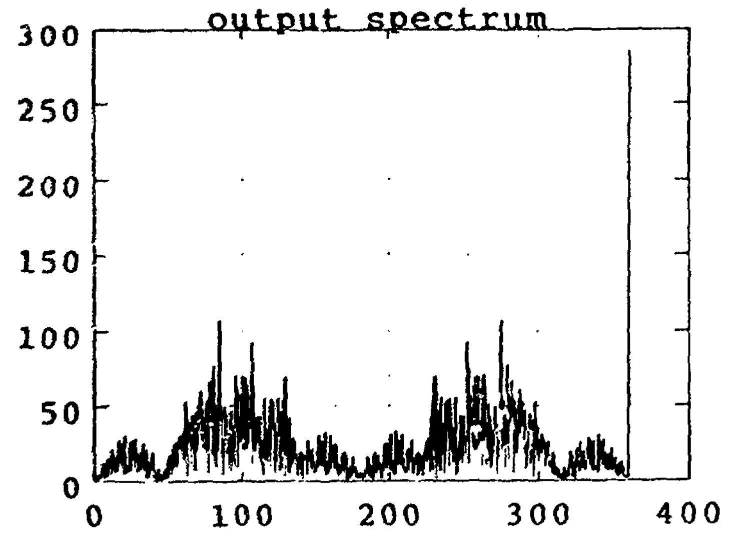


Fig 4.21

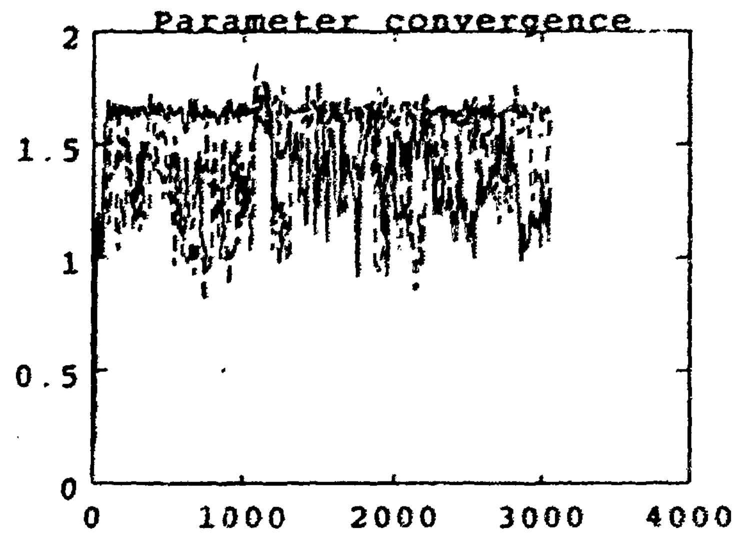


Fig 4.22

```
Model mismatch m=1 and n=3
Input ...................DS-BPSK
Number of interferers ...2
Samples per carrier cycle of the
interferes .............12.0,3.0
Samples per Chip bit of
interferers ............200,200,200
Broad band signal samples per
carrier cycle ..........4.0
Broad band signal samples per
chip bit ...............8.0
Adaptive filter step ....0.02
Adaptive filter r .......0.9
```

TABLE 4.1

Case	*# of sections*	*Sines*	v_{min}	v_{max}	*Noise*	*zero order*		*1st order*		*2nd order*	
						μ	*iter*	μ	*iter*	μ	*iter*
1a	1	1	0.05	20	no	0.05	67	0.05	20	0.05	20
1an	1	1	0.05	20	yes	0.05	jump	0.05	20	0.05	144
1b	1	3	0.05	20	no	0.05	jump	0.05	30	0.05	25
1bn	1	3	0.05	20	yes	0.05	jump	0.05	50	0.05	100
2a	5	3	0.10	10	no	0.017	340	0.063	200	0.07	400
2an	5	3	0.10	10	yes	0.017	1200	0.063	400	0.07	450
2b	5	5	0.10	10	no	0.017	1410	0.063	150	0.07	500
2bn	5	5	0.10	10	yes	0.017	3500	0.063	175	0.07	250
2c	5	7	0.10	10	no	0.017	1667	0.063	450	0.07	jump
2cn	5	7	0.10	10	yes	0.017	700	0.063	167	0.07	jump
3a	10	7	0.02	50	no	0.02	1667	0.075	500	0.07	1500
3an	10	7	0.02	50	yes	0.02	5000	0.075	750	0.07	1000
3b	10	10	0.02	50	no	0.02	7500	0.075	2000	0.07	5250
3bn	10	10	0.02	50	yes	0.02	1000	0.075	2000	0.07	2500

- In all cases with noise, parameters converge to exact value when number of notches is greater or equal to number of sinusoids. Iterations listed in table represent convergence to three decimal places.

- In all cases with noise, parameter oscillate around the exact value. The number of iterations listed in the table is the number of iterations required to establish this oscillatory pattern.

- In some cases with more interfering sine waves than notches will jump at random between sine waves. This pattern is indicated in the table by the word *jump*.

- Values of v_{min} and v_{max} in the table refer only to zero-order forgetting which generally will not converge without limits on v. All other types of forgetting were run without limits on v

V. HARDWARE IMPLEMENTATION

Although hardware design is beyond the scope of this thesis, in this chapter we shall briefly look at possible hardware configurations with a view toward feasibility of the new algorithm.

A. HARDWARE FEASIBILITY

In recent years many dedicated digital signal processing chips have become commercially available. Table 5.1 gives characteristics of some typical chips that are currently available.

TABLE 5.1

Feature	*Commercial Make*			
	MIPS R3010	**Weitek 3364**	**TI 8847**	**AT&T DSP32c**
Cycle Time(ns)	40	50	30	20
Cycles/add	2	2	2	2
Cycles/mult	5	2	3	2
Cycles/divide	19	17	11	3
Cycles/sq root	-	30	14	?

Data on MIPS R3010, Weitek 3364, and TI 8847 was obtained from the computer architecture book by Hennesey and Patetrson [Ref. 30] while the data for AT&T DSP32c is obtained from DSP32c data manual [Ref. 31]. Fig 5.1. gives a schematic of the 2nd order IIR filter hardware scheme. It has a coefficient memory which can be set by an external device. Close examination reveals that *Multiplications A* can be performed simultaneously while *Multiplication B* and other additions are to be in sequence. Fig 5.2. is the basic building block for the adaptive filter and is identical for all the sections. This is offers an advantage in hardware complexity over the Kwan & Martin approach. Filters 1 and 2 have transfer functions of $H_{bp}(z)$ while filter 3 has a

extrapolated data.

2nd Order IIR Filter

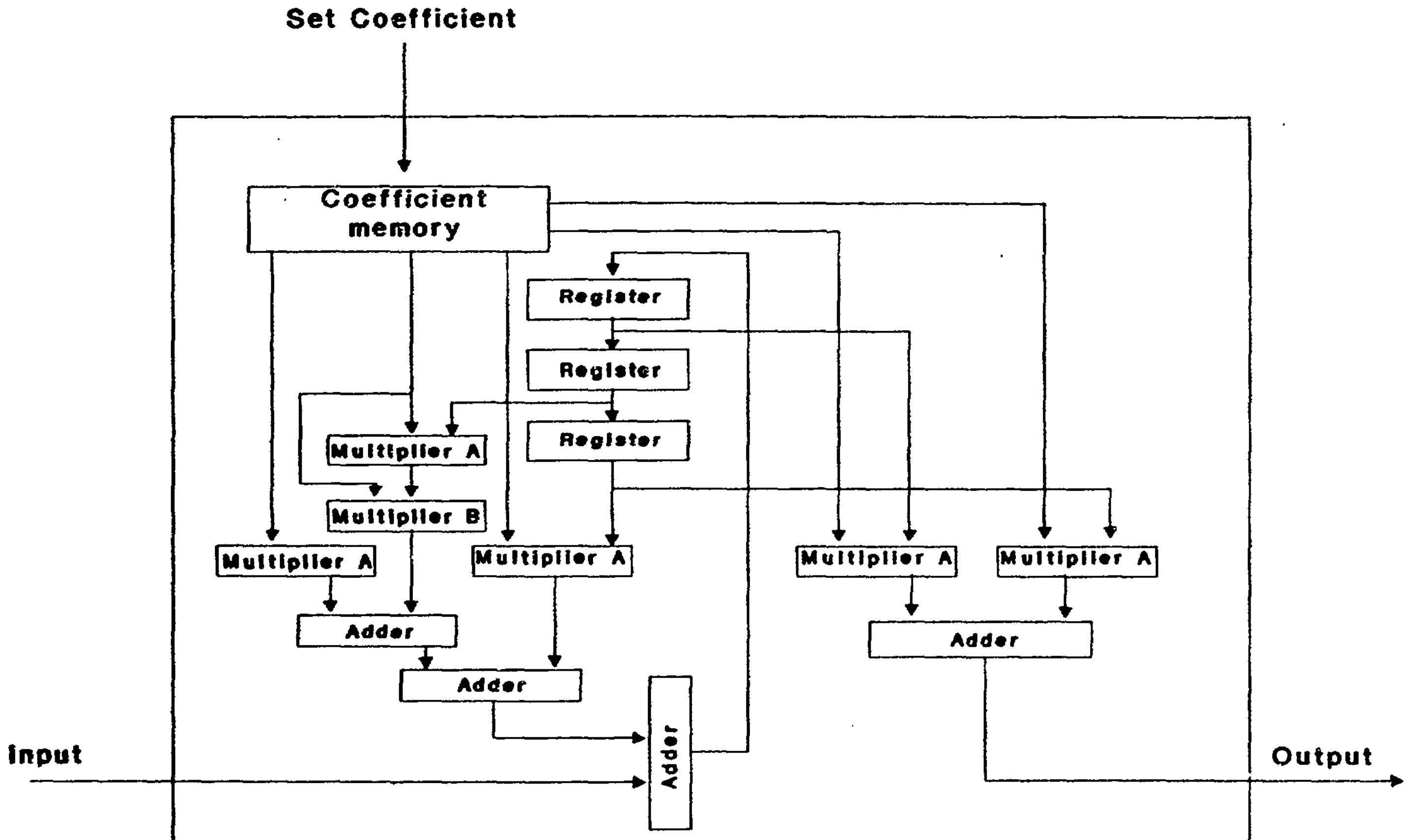
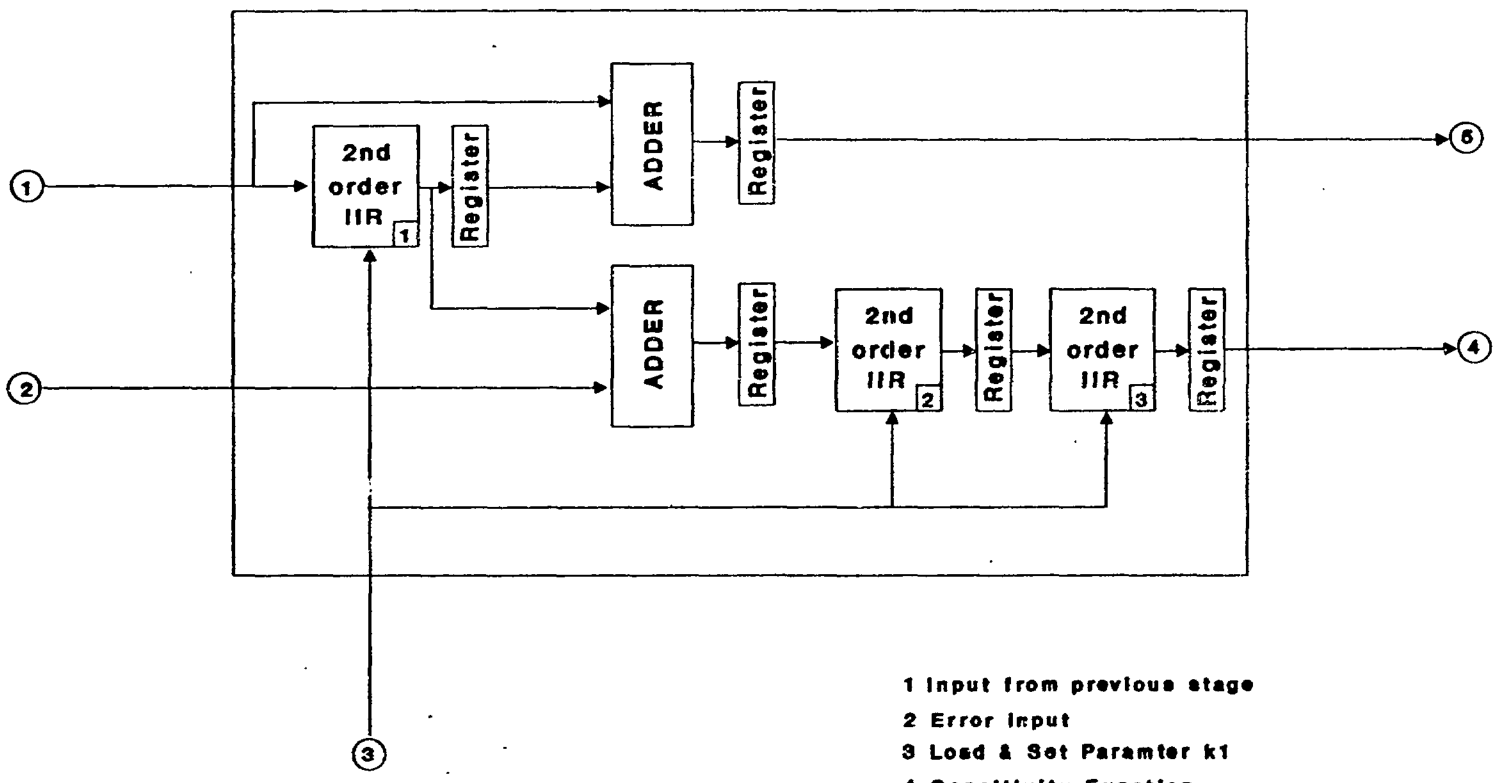


Fig 5.1

Basic Building Block



1 Input from previous stage
2 Error Input
3 Load & Set Paramter k1
4 Sensitivity Function
5 Output to next stage

Fig 5.2

transfer function $H_{ss}(z)$. Throughput rate is primarily determined by the computing time of the 2nd order IIR filter. The overall block diagram of the adaptive filter for a 3-Notch cancellation was given in Fig 5.3. This architecture remains same either for a Floating point or Fixed Point arithmetic.

1. Time Budget

In the architecture of Fig 5.1, 5.2, and 5.3 the most vital elements are the *IIR filter*, *Controller* and the *Forgetting filter*. The IIR filter corresponds to $H_{bp}(z)$. The controller has to update the parameter via equation (3.17) which calls for 2 multiplications, 1 addition, and 1 division. Similarly the Forgetting Filter has to implement equation (3.19) calling for 2 multiplications and 1 addition. Results are tabulated as given below

TABLE 5.2

Element	# of Effective *Multiplications*	*# of Effective Additions*	*# of Divisions*
IIR filter	2	3	nil
Controller	2	1	1
Fogetting Filter	2	1	nil

The maximum computing time is at the IIR filter. Using the AT&T DSP32c processor, it takes $5 \times 2 = 10$ cylces (*ie* $200ns$) for implementation the desired IIR filter. This corresponds to a Throughput rate of $5Mhz$. This is the best that could be achieved with the existing commercial floating point processors. However processors like the HDSP66110 of Honey well make claim an ALU speed of $10ns$ with *block floating point* operations can conveniently implement at $10Mhz$ throughput rate, without pipelining the IIR filter. With pipelining the throughput rate would increase dramatically, but will never exceed the maximum throughput rate of $100Mhz$.

3-Notch Canceller

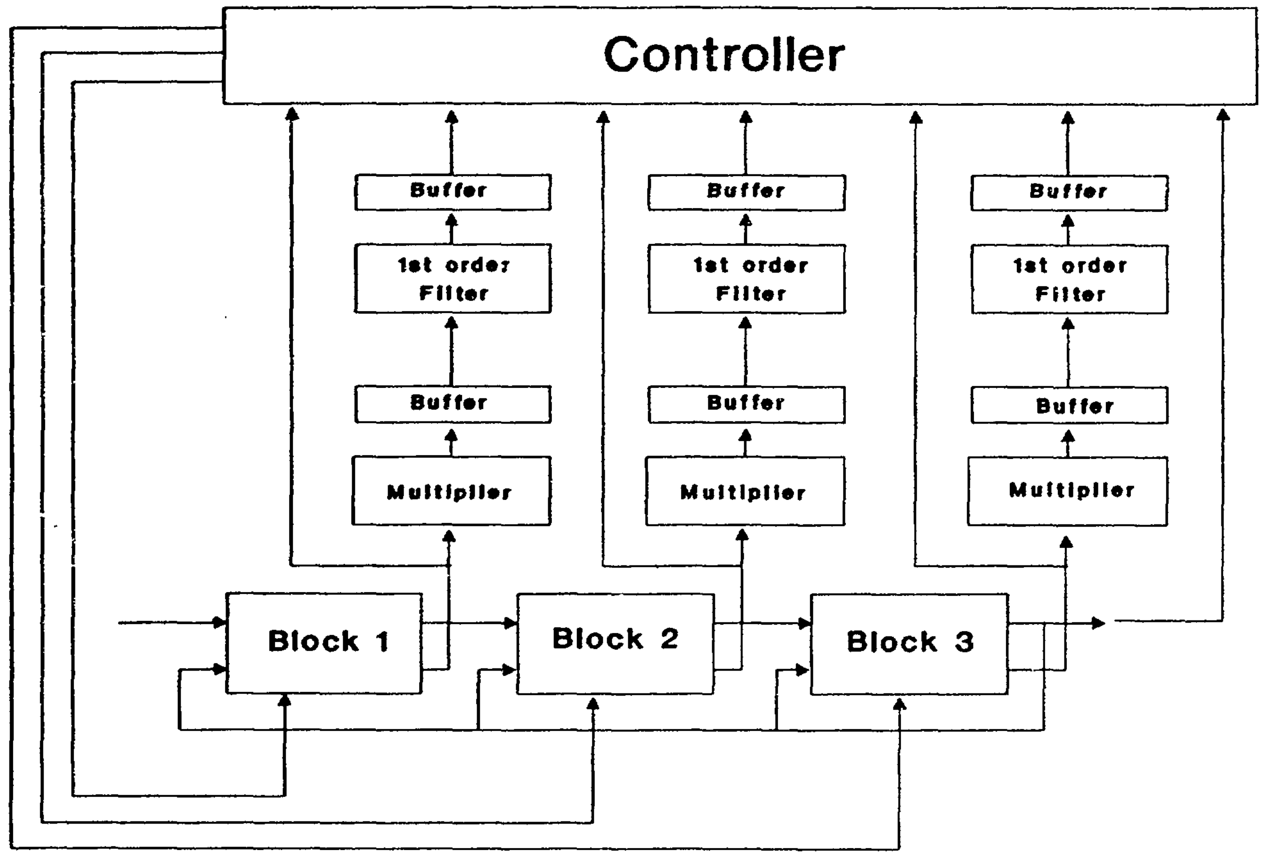


Fig 5.3

2. VLSI Approach

Taking advantage of the fact that the entire hardware can be generated by a simple repetitive use of the building block, strong consideration should be given to designing a VLSI chip for Fig 5.2 rather than implementing the hardware via the existing commercial chips. The main rationale behind this idea is that reliability of the system is very important in EW equipment. Also VLSI has the potential of ultimate-low cost.

VI. CONCLUSIONS

In this thesis the problem of suppressing narrow-band interference was addressed. The problem specific to the Electronic warfare scenario was kept in mind while solving the problem. The new algorithm [Ref. 26] was derived and simulated. This new algorithm was tested against various signals and signal conditions. The results are highly encouraging.

Some aspects of pipelining were discussed. Reduced hardware complexity of the New Algorithm is an important advantage over the earlier algorithm by Kwan and Martin [Ref. 8]. A possible hardware scheme for implementing the new algorithm was discussed. It was observed that with the existing commercially available floating point processors a throughput rate of $5Mhz$ is achievable while using processors like HDSP66110 with an ALU speed of $10ns$ it is possible to achieve a throughput rate of $10Mhz$.

A. FUTURE WORK

Future directions of work are

i) VLSI design of the system with an architecture
as indicated earlier or a similar architecture.

ii) Pipelining of 2nd Order IIR filter suited for
this application

iii) Design using recursive DFT

The above areas of work are the logical extensions of this work. However a radically different approach for the same problem using adaptive filtering in frequency domain [Ref. 10] should also be considered.

APPENDIX A

```
      dimension faray(5000)
      dimension uaray(5000),yaray(5000)
      dimension fhat(5000),ph(5000)
      dimension a(90),b(90)
      open (unit=9,file='spk.dat',status='new')
c
c     This programme generates input output
c     sequence by exciting a linear system
c     defined by the numerator polynomial
c     B(z) and denominator polynomial A(z)
c     this data is stored in uarray and yarray
c     Subsequently an ARMA (pole,zero) is used
c     to fit this data.
c
      n=1024
      ix=1
      yk=rand(ix)
      ix=0
      kk=8
c
c     generate sequence uk and yk
c
      ip=3
      iz=2
      a(1)=1.7
      a(2)=-1.53
      a(3)=0.648
      b(1)=1.0
      b(2)=0.6
      do 10 k=1,n
      call rbn(uk,ix,k)
      call system(a,b,uk,yk,ip,iz,k)
      uaray(k)=uk
      yaray(k)=yk
10    continue
c
c     save the sequence in the arrays uaray and yaray
c
      call  spktrm(yaray,faray)
c
c     spectrum of the output sequence yaray is computed
c     using FFT programme and stored in faray for
c     comparision
c
      ip=4
      iz=1
c
c     Fit an ARMA(ip,iz) model for the given data
c     this is an ordinary least squares algorithm
```

```
c
      print *,'input-ip-iz-->'
      read *,ip,iz
      do 20 k=1,n
      uk=uaray(k)
      yk=yaray(k)
      call arma(a,b,uk,yk,ip,iz,kk,k)
c     print *,'num->',(b(i),i=1,iz)
c     print *,'den->',(a(i),i=1,ip)
20    continue
      print *,'num->',(b(i),i=1,iz)
      print *,'den->',(a(i),i=1,ip)
c
c     Sampling of the Z-transform. given
c     coefficients a .... and b .....
c     corresponding H(z) = B(z)/A(z)
c     is evaluated on the unit circle
c     for obtaing the parametric spectrum
c
      call zsmple(a,b,fhat,ph,ip,iz)
c
c


      do 30 i=1,n/2
c     print *,i,'-->',fhat(i),faray(i)
      write (9,*) i,fhat(i),faray(i)
30    continue
      stop
      end
c
c
      subroutine zsmple(a,b,r,ph,ip,iz)
      dimension a(90),b(90)
      dimension r(2048),ph(2048)
      complex z,ui,fn,fd,omega,delw,spec
      pi=atan(1.0)
      pi=4.0*pi
      ui=(0.0,1.0)
      delw=(pi/1024.0)*ui*2.0
      omega=(0.0,0.0)
      do 100 k=1,1024
      z=exp(-omega)
      fd=(0.0,0.0)
      do 10 i=1,ip
10    fd=fd+a(i)*(z**(-i))
      fd=1.0-fd
      fn=(0.0,0.0)
      do 20 i=1,iz
20    fn=fn+b(i)*(z**(-i))
      spec=fn/fd
      r(k)=abs(spec)
      specr=real(spec)
      speci=aimag(spec)
      ratio=speci/specr
      ph(k)=atand(ratio)
      omega=omega+delw
c     print *,'--z-om->',z,omega
100   continue
      return
      end
c
c
      function atand(x)
      pi=atan(1.0)
      pi=4.0*pi
      xrad=atan(x)
      atand=xrad*(180.0/pi)
      return
      end
c
c
      subroutine rbn(b,ix,k)
      z=rand(ix)-0.5
      if(z.ge.0.0) b=1.0
      if(z.lt.0.0) b=-1.0
      return
      end

c
c
         subroutine system(a,b,uk,yk,ip,iz,k)
         dimension
     &    zkar(90),zkma(90),
     &    a(90),b(90)
      if (k.ne.1) go to 250
      do 230 i=1,ip
230   zkar(i)=0.0
      do 240 i=1,iz
240   zkma(i)=0.0
250   continue
      do 30 i=ip,2,-1
30    zkar(i)=zkar(i-1)
      zkar(1)=yk
      do 31 i=iz,2,-1
31    zkma(i)=zkma(i-1)
      zkma(1)=uk
c
c
      yk=0.0
      do 200 i=1,ip
200   yk=yk+a(i)*zkar(i)
      do 210 i=1,iz
210   yk=yk+zkma(i)*b(i)
         return
```

```
          end
c
c
      subroutine delay(uk,yk,idelay,k)
      dimension d(500)
      if (k.ne.1) goto 10
      do 20 i=1,500
20    d(i)=0.0
10    continue
      do 30 i=500,2,-1
30    d(i)=d(i-1)
      d(1)=uk
      yk=d(idelay+1)
      return
      end
c
c
      subroutine spktrm(taray,faray)
      complex u(5000)
      dimension taray(5000),faray(5000)
      m=10
      n=1024
      rn=n
      do 20 i=1,n
      u(i)=taray(i)
20    continue
      call pfft(u,m,n)
      do 30 i=1,n/2
      faray(i)=real(u(i))
30    continue
      return
      end
c
c
c     **********************************
c                              *
c     Subroutine for computing DFT of      *
c     an array 'a' is complex and a pair
c     cf numbers are to be specified       *
c     for each point           *
c     m is the 2 power index       *
c     say m=10 then number of points   *
c     are 1024                 *
c     after computation fft is kept    *
c     in the same complex array 'a'    *
c                              *
c     **********************************
c
      subroutine pfft(a,m,n)
      complex a(5000),u,w,t
      complex ui,ur
      pi=3.141592653589793
      n=2**m
      nv2=n/2
      nm1=n-1
      j=1
      do 10 i=1,nm1
      if(i.ge.j) goto 20
      t=a(j)
      a(j)=a(i)
      a(i)=t
20    k=nv2
30    if(k.ge.j) goto 10
      j=j-k
      k=k/2
      goto 30
10    j=j+k
      do 40 l=1,m
      le=2**l
      le1=le/2
      u=(1.0,0.0)
      w=cmplx(cos(pi/le1),-sin(pi/le1))
      do 40 j=1,le1
      do 50 i=j,n,le
      ip=i+le1
      t=a(ip)*u
      a(ip)=a(i)-t
50    a(i)=a(i)+t
40    u=u*w
      rn=n/2
      ui=(0.0,1.0)
      ur=(1.0,0.0)
      do 60 i=1,n/2
      a(i)=a(i)/rn
      a(i+512)=a(i+512)/rn
      areal=real(a(i))
      aimgn=aimag(a(i))
      amagn=abs(a(i))
      aphase=atand(aimgn/areal)
      a(i)=ur*amagn+ui*aphase
      areal=real(a(i+512))
      aimgn=aimag(a(i+512))
      amagn=abs(a(i+512))
      aphase=atand(aimgn/areal)
      a(i+512)=ur*amagn+ui*aphase
60    continue
      return
      end
c
c
c     $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
c
c
         subroutine arma(a,b,uk,yk,ip,iz,kk,k)
         dimension
      &    zkar(90),zkma(90),
      &    zk(90),a(90),b(90),c(90)
      if (k.ne.1) go to 250
      n=ip+iz
      ipp=ip+1
      do 230 i=1,ip
230   a(i)=0.1
      do 240 i=1,iz
240   b(i)=0.1
250   continue
      do 30 i=ip,2,-1
30    zkar(i)=zkar(i-1)
      zkar(1)=ykold
      ykold=yk
      do 31 i=iz,2,-1
31    zkma(i)=zkma(i-1)
      zkma(1)=uk
      do 32 i=1,ip
      c(i)=a(i)
32    zk(i)=zkar(i)
      do 33 i=1,iz
      c(i+ip)=b(i)
33    zk(i+ip)=zkma(i)
      call syseq(c,yk,zk,n,kk,k)
      do 34 i=1,ip
34    a(i)=c(i)
      do 35 i=1,iz
35    b(i)=c(i+ip)
         return
      end
c
c
c
         subroutine syseq(a,yk,zk,n,kk,k)
c
c     ****************************************
c     solves system of equations with
c     # of equations more than unknowns
c     using linear reggression.
c                 t
c     equation is  yk=zk * a
c     where 'a' is n vector to be estimated.
c
c     *****************************************
c
         dimension
      &    del(90),phat(90,90),zk(90),y(90),a(90)
      if (k.ne.1) go to 250
      alpha=1.0
250   continue
         call inv(phat,zk,alpha,n,kk,k)
      ykhat=0.0
      do 40 i=1,n
40    ykhat=ykhat+zk(i)*a(i)
      er=yk-ykhat
         do 60 i=1,n
60    y(i)=er*zk(i)
         do 80 i=1,n
         del(i)=0.0
         do 80 j=1,n
80    del(i)=phat(i,j)*y(j)+del(i)
         if(mod(k,kk).ne.0) goto 70
c     print *,'--->',k
c     print 100, ((phat(i,j),i=1,n),j=1,n)
c     print 130, (del(i),i=1,n)
c     print 160,er
c     print 170, (zk(j),j=1,n)
70    continue
      do 50 i=1,n
50    a(i)=a(i)+del(i)
100   format(2x,'phat',2x,/,4e16.6)
120   format(2x,'zk',2x,/,4e16.6)
130   format(2x,'del',2x,/,4e16.6)
160   format(2x,'ek1',2x,e16.6,/)
170   format(2x,'zk',2x,/,4e16.6)
         return
      end
c
c
         subroutine inv(phat,zk,alpha,n,kk,k)
         dimension phat(90,90),zk(90),q(90),r(90),delp(90,90)
         if(k.ne.1) goto 5
         do 6 i=1,n
         do 6 j=1,n
         if(i.eq.j) phat(i,j)=1.0
         if(i.ne.j) phat(i,j)=0.0
6     continue
5     continue
         do 90 i=1,n
         do 90 j=1,n
         phat(i,j)=phat(i,j)/alpha
90    continue
         do 10 i=1,n
         q(i)=0.0
         do 10 j=1,n
10    q(i)=phat(i,j)*zk(j)+q(i)
      sum=1.0
         do 20 i=1,n
20    sum=sum+zk(i)*q(i)
         do 30 j=1,n
         r(j)=0.0
         do 30 i=1,n
30    r(j)=phat(i,j)*zk(i)+r(j)
```

```
      do 40 j=1,n
      do 40 i=1,n
40    delp(i,j)=q(i)*r(j)
      do 50 j=1,n
      do 50 i=1,n
50    phat(i,j)=phat(i,j)-(delp(i,j)/sum)
      return
      end
```

APPENDIX B

```
c
c    simulation of michal paper IEEE
c
     dimension ak1(10),ak2(10),s(10),xout(10),theta(10)
     dimension aks(10),wp(10),ak3(10)
     dimension array(4,4000)
     open( unit=9,file='martin.dat',status='new')
     open( unit=8,file='mart.dat',status='new')
     print *,'-input-SNR-in-dB--->'
     read *,snr
c    if (snr.ne.0) goto 200
c    var=0.5
c    goto 210
     var=1.0/(2.0*(10.0**(snr/10.0)))
210  continue
     print *,'--variance-->',var
     n=1
     print *,'-input-nh-->'
     read *,n
     print *,'-input-#-of-waves-->'
     read *,nd
     print *,'-input-angles-->'
     read *,(theta(i),i=1,n)
     r=0.9
     pi=4.0*atan(1.0)
     rad=pi/180.0
     amu=0.0
c    do 50 kl=1,50
     amu=0.015
     print *,'-input-step-size-- 0.015-->',amu
     read *,amu
     print *,'--angles-->',(theta(i),i=1,n)
     print *,'--input--1-for-step-change--else-0-->'
     read *,ic
     if (ic.eq.1) print *,'--input-step-in-degrees-->'
     if (ic.eq.1) read *,step
     do 40 i=1,n
     ak2(i)=(1-r*r)
     ux=theta(i)*rad
     value=cos(ux)
c    ak1(i)=d1*sqrt(1.0+r*r-2.0*r*value)
     ak1(i)=sqrt(1.0+r*r)
     aks(i)=sqrt(1.0+r*r-2.0*r*value)
40   continue
c    amu=amu+0.0001
     do 10 k=1,6500
     call system(yk,wk,theta,step,var,ic,nd,k)
     call filters(ak1,ak2,aks,s,en,xout,yk,n,k)
     call increment(ak1,ak2,aks,s,en,amu,avg,n,k)
     do 30 i=1,n
```

```
      xx=2.0*sqrt(1-0.5*ak2(i))
      wp(i)=2.0*asin(ak1(i)/xx)
c     wp(i)=wp(i)/rad
      wp(i)=cos(wp(i))
30    continue
      do 60 i=1,n
      ak3(i)=abs(ak1(i))
60    continue
      kk=mod(k,100)
      ll=mod(k,10)
      if(kk.eq.0) print *,'--ak1-->',(ak3(j),j=1,n)
      if(ll.eq.0) write(9,*) k,(ak3(j),j=1,n)
      if(kk.eq.0) print *,'--aks-->',(aks(j),j=1,n)
c     if(kk.eq.0) print *,k,'-avg->',avg
c     print *,(wp(i),i=1,n)
c     write(9,*) k,(wp(i),i=1,n)
      if(k.gt.1500.and.k.lt.2500) write(8,*) k,yk,en
      if(k.gt.5000.and.k.lt.6000) write(8,*) k,yk,en
10    continue
c     print *,   kl,amu,avg
c     write(9,*) kl,amu,avg
50    continue
      close(9)
      stop
      end
c
c
      subroutine increment(ak1,ak2,aks,s,en,amu,avg,n,k)
      dimension ak1(10),ak2(10),aks(10),s(10),ss(10)
      dimension dec(10),flag(10),fading(10)
      if (k.ne.1) goto 30
c     amu=0.001
      avg=0.0
      fading(1)=0.9
      fading(2)=0.9
      fading(3)=0.9
      fading(4)=0.9
30    continue
      do 10 i=1,n
      ss(i)=fading(i)*ss(i)+s(i)*s(i)*(1.0-fading(i))
10    continue
c     do 31 i=1,n
c31   print *,'-en-s-ss->',en,s(i),ss(i)
c     sum=0.0
      do 20 i=1,n
      dec(i)=amu*en*s(i)/(ss(i)+0.0001)
      ak1(i)=ak1(i)-dec(i)
c     sum=sum+(abs(aks(i)-ak1(i)))/(aks(i))
      call stability(ak1,ak2,flag,n)
      ak1(i)=ak1(i)+flag(i)*dec(i)
20    continue
      realk=k
      avg=avg*((realk-1.0)/realk)+sum/realk
      return
      end
c
c
c
      subroutine filters(ak1,ak2,aks,s,en,xout,xin,n,ki)
      dimension w(10,3),a(10,2),b(10,2),e(11),xbp(10)
      dimension g(10,3),gbp(10),u(10)
      dimension q(10,3),s(10),xout(10),c(10)
      dimension ak1(10),ak2(10),aks(10),as(10)
c
      if (ki.ne.1) goto 200
c     open( unit=9,file='martin.dat',status='new')
200   continue
      i=n
      do 40 i=1,n
      a(i,1)=-(ak1(i)*ak1(i)+ak2(i)-2.0)
      as(i)=-(aks(i)*aks(i)+ak2(i)-2.0)
      a(i,2)=-(1.0-ak2(i))
      b(i,1)=-0.5*ak2(i)
      b(i,2)=0.5*ak2(i)
      c(i)=-ak1(i)*ak2(i)
40    continue
c     do 41 i=1,n
c     print *,'-k1-k2->',ak1(i),ak2(i)
c41   print *,'-a-b-c->',a(i,1),a(i,2),b(i,1),b(i,2),c(i)
c
c
      e(1)=xin
      i=n
      do 10 j=1,n
      w(i,3)=w(i,2)
      w(i,2)=w(i,1)
      w(i,1)=a(i,1)*w(i,2)+a(i,2)*w(i,3)+e(j)
      xbp(j)=b(i,1)*w(i,1)+b(i,2)*w(i,3)
      e(j+1)=xbp(j)+e(j)
      i=i-1
10    continue
      en=e(n+1)
c
c
       i=n
      do 20 k=1,n
       u(k)=en-xbp(k)
           g(i,3)=g(i,2)
           g(i,2)=g(i,1)
           g(i,1)=a(i,1)*g(i,2)+a(i,2)*g(i,3)+u(k)
           gbp(k)=b(i,1)*g(i,1)+b(i,2)*g(i,3)
       q(k,3)=q(k,2)
       q(k,2)=q(k,1)
       q(k,1)=a(i,1)*q(k,2)+a(i,2)*q(k,3)+gbp(k)
      s(i)=c(i)*q(k,2)
      i=i-1
20    continue
      kk=mod(ki,10)
c     if(kk.eq.0) write(9,*) ki,en,a(1,1),as(1),a(2,1),as(2)
c     if(kk.eq.0) print *,   ki,(a(i,1),i=1,n)
c     if(kk.eq.0) print *,   ki,(as(i),i=1,n)
c     if(ki.ge.1800) write(9,*) ki,e(1),en,xbp(1),a(1,1),as(1)
c     if(ki.ge.1800) print *,   ki,e(1),en,xbp(1),a(1,1),as(1)
      return
      end
c
c
c
      subroutine system(yk,wk,theta,step,var,ic,n,k)
      dimension theta(10),yout(10),yz(10)
      if (k.ne.1) goto 100
100   continue
c     call bpf(theta,yz,n,k)
c     call waves(theta,yout,n,k)
      call file(yk,theta,step,ic,n,k)
      sum=0.0
      wk=0.0
      do 10 i=1,n
      sum=sum+yout(i)
      wk=wk+yz(i)
10    continue
      call gnoise(g,var,k)
c     yk=sum+g
c     yk=wk+g
      yk=yk+g
      return
      end
c
c
      subroutine waves(theta,yout,n,k)
      dimension zkar(10,3),theta(10),yout(10),a1(10)
      if(k.ne.1) goto 10
      pi=(atan(1.0))*4.0
      rad=pi/180.0
      do 20 i=1,n
      zkar(i,1)=0.0
      zkar(i,2)=-sin(theta(i)*rad)
      a1(i)=2.0*cos(theta(i)*rad)
20    continue
10    continue
      do 30 i=1,n
      zkar(i,3)=zkar(i,2)
      zkar(i,2)=zkar(i,1)
      zkar(i,1)=a1(i)*zkar(i,2)-zkar(i,3)
      yout(i)=zkar(i,1)
30    continue
```

```
      return
      end
c
c
      subroutine gnoise(g,s,k)
      if(k.ne.1) goto 10
      ix=1
      yfl=rand(ix)
      ix=0
10    continue
      sum=0.0
      do 20 i=1,12
      yfl=rand(ix)
20    sum=sum+yfl
      g=(sum-6.0)*s
      return
      end
c
c
      subroutine stability(ak1,ak2,flag,n)
      dimension ak1(10),ak2(10),flag(10),a(10,2)
      do 30 i=1,n
      flag(i)=0.0
      a(i,1)=-(ak1(i)*ak1(i)+ak2(i)-2.0)
      a(i,2)=-(1.0-ak2(i))
      des=a(i,1)*a(i,1)+4.0*a(i,2)
      if (des.lt.0.0) goto 10
      des=0.5*sqrt(des)
      root1=0.5*a(i,1)+des
      r1=abs(root1)
      root2=0.5*a(i,1)-des
      r2=abs(root2)
      if (r1.ge.1.0) goto 20
      if (r2.ge.1.0) goto 20
c     print *,'-real-roots-->',root1,root2
      goto 30
20    flag(i)=1.0
      goto 30
10    des=-des
      y=0.5*sqrt(des)
      x=0.5*a(i,1)
      xx=abs(x)
      if (xx.le.1.0e-6) x=0.0000001
      angle=(atan(y/x))*57.3
      radius=sqrt(x*x+y*y)
c     print *,'--complex--roots-->',radius,angle
      if(radius.ge.1.0) goto 20
30    continue
      return
      end
c
c
```

```
c     $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
c
c
      subroutine bpf(theta,yout,n,k)
         dimension zkar(10,3),a(10,3),zkma(10,4),b(10,3)
      dimension theta(10),g(10),yin(10),yout(10),uk(10)
      if (k.ne.1) go to 10
      pi=4.0*atan(1.0)
      rad=pi/180.0
      do 15 i=1,n
      zkar(i,1)=0.0
15    zkar(i,2)=0.0
      r=0.985
c     print *,'--input--r-->',r
c     read *,r
      do 25 i=1,n
      a(i,1)=1
      a(i,3)=r*r
      g(i)=0.5*(1-r*r)
      b(i,1)=g(i)
      b(i,3)=-g(i)
      a(i,2)=2*r*cos(theta(i)*rad)
25    b(i,2)=0.0
10    continue
      save=var
      do 35 i=1,n
      var=2.0
      call gnoise(gk,var,k)
      yin(i)=gk
      zkar(i,3)=zkar(i,2)
      zkar(i,2)=zkar(i,1)
      zkma(i,3)=zkma(i,2)
      zkma(i,2)=zkma(i,1)
      zkma(i,1)=yin(i)
      uk(i)=zkma(i,1)*b(i,1)-zkma(i,2)*b(i,2)+zkma(i,3)*b(i,3)
      zkar(i,1)=zkar(i,2)*a(i,2)-zkar(i,3)*a(i,3)+uk(i)
c     print *,'--fil-->',afil,theta,r
c     print *,'--fil-->',zkar
35    yout(i)=zkar(i,1)
      var=save
         return
         end
c     $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
c
      subroutine file(yk,theta,step,ic,n,k)
      dimension output(8000),theta(10)
      if (k.ne.1) goto 10
      call gendata(output,theta,step,ic,n,k)
10    continue
      yk=output(k)
      return
      end
```

```
c     $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
c     $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
c     $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$



c
      subroutine gendata(output,theta,step,ic,nn,k)
      DIMENSION YSAMP(65536),data(4,8000),output(8000)
      dimension theta(10)
c     OPEN (UNIT=9,FILE='bpsk.dat',STATUS='NEW')
      n=7000
      amag=1.414
      tcarr=4.8
      tchip=7.0
      ichip=0
      tdata=n
      tdelay=0.0
      l=9999
      tchip=200.0
      do 10 i=1,nn
      tcarr=360.0/theta(i)
      call bpsk(n,amag,tcarr,tchip,ichip,tdata,tdelay,l,ysamp)
      do 20 j=1,n
      if (ic.eq.0) goto 80
      if (i.eq.2) goto 50
      data(i,j)=ysamp(j)
      goto 60
50    if (j.gt.3000) ysamp(j)=0.0
80    data(i,j)=ysamp(j)
60    continue
20    continue
      print *,i,'-->',theta(i),tcarr
10    continue
      tcarr=360.0/90.0
      tchip=8.0
      call bpsk(n,amag,tcarr,tchip,ichip,tdata,tdelay,l,ysamp)
      do 30 i=1,n
      x=0.0
      do 40 j=1,nn
40    x=x+data(j,i)
      output(i)=x+ysamp(i)
c     write(9,*) x
30    continue
      if (ic.eq.0) return
      n1=n-3000
      tchip=200.0
      theta1=theta(2)+step
      tcarr=360.0/theta1
      call bpsk(n1,amag,tcarr,tchip,ichip,tdata,tdelay,l,ysamp)
      j=1
      do 70 i=3000,n
      output(i)=output(i)+ysamp(j)
      j=j+1
70    continue
c     close(9)
      return
      end
c
c
c     this programme was modified to suit the martin
c     programme
      subroutine bpsk
      &    (n,amag,tcarr,tchip,ichip,tdata,tdelay,l1,ysamp)
c
c
c
C     THIS PROGRAM GENERATES SAMPLES OF A DIRECT SEQUENCE BI-PHASE
SHIFT
C     KEYED SPREAD SPECTRUM SIGNAL.  THE "INFORMATION" BITS AND THE
C     SPREADING SEQUENCE USED ARE RANDOMLY GENERATED.  PARAMETERS
REQUIRED
C     FOR OPERATION ARE:
C                     N    =  NUMBER OF SAMPLES GENERATED
C                            FOR CONSISTENCY WITH USE BY AN FFT
C                            ALGORITHM, N SHOULD BE AN INTEGER
C                            POWER OF 2 -- TYPICALLY 1024
C                              NOTE: IF N>1024 DIM OF YSAMP MUST CHANGE
C                MAG       =  MAGNITUDE OF CARRIER WAVEFORM
C                TSDELAY   =  NUMBER OF SAMPLE TIMES DELAYED FROM
C                               t = 0 BEFORE BEGINNING SAMPLES
C                            AN ARBITRARY VALUE THAT ALLOWS SOME
C                            FLEXIBILITY IN SAMPLING.  SHOULD BE
C                            ZERO FOR SAMPLING AT t=0.
C                TDATA     =  NUMBER OF SAMPLE TIMES IN ONE DATA BIT
C                TCHIP     =  NUMBER OF SAMPLE TIMES IN ONE BIT OF
C                               SPREADING CODE
C                TCARR     =  NUMBER OF SAMPLE TIMES IN ONE CYCLE OF
C                            CARRIER FREQUENCY
C                TSAMP     =  DURATION OF A SAMPLE TIME.  IN GENERAL
C                                 TSAMP WILL ALWAYS BE = 1.0, SINCE
C                            TIME SAMPLED VALUES BECOME STATIC
C                            WHEN STORED AND CAN BE SCALED LATER.
C
C     THE ALGORITHM USED TO GENERATE THE DS-BPSK SIGNAL RELIES UPON
THE
C     BUILTIN FORTRAN RANDOM NUMBER GENERATOR RAND(L) TO PRODUCE THE

C     INFORMATION BIT STREAM AND THE SPREADING SEQUENCE CODE.  AT
THE TIME
C     OF EACH SAMPLE, THESE TWO BITS ARE MODULO 2 ADDED TO PRODUCE
A CHIP
C     BIT.  THIS BIT IS THEN USED TO SHIFT THE PHASE OF THE CARRIER
BY
```

```
C     180 DEGREES IF 1 OR BY 0 DEGREES IF 0.  THE RESULTANT VALUE OF
THE
C     COSINE  FUNCTION  IS  MULTIPLIED  BY  THE  AMPLITUDE  OF  THE
WAVEFORM, THEN
C     STORED IN FILE BT.DAT IN THIS FILE DIRECTORY.  THE PROCESS IS
CONTIN-
C     UED UNTIL N SAMPLES ARE GENERATED.  NOTE THAT FIRST LINE OF
BT.DAT
C     CONTAINS N, DSBPSK, DATE AND TIME AS ID FIELD FOR FILE.
C
C                                W.R.TUCKER 9-MAY-83
C          CONVERTED PROGRAM TO ALLOW FOR M-SEQUENCE CHIP
C          M-SEQUENCE GENERATOR REQUIRES DATA FILE FSR.DAT WITH
C          PARAMETERS FOR FEEDBACK SHIFT REGISTER.
C                                W.R.TUCKER 4-AUG-83
C
C     INITIALIZE -- I = INFO BIT NUMBER, J = SP SEQ CODE BIT NUMBER
C     L IS ARGUMENT FOR RAND(L)--A DIFFERENT SEQUENCE MAY BE
C     GENERATED BY INITIALIZING L WITH DIFFERENT VALUES.
C
C     Modified by HHL for installation on ULTRIX ECE VAX, 12/90
c     Writes output in file bpsk.dat
c
      DIMENSION YSAMP(65536)
      INTEGER I,J,L,INFO,CHIP,IFSR,IFBD(100),IVAL(100)
      REAL TDATA,TCHIP,TCARR,TDELAY,TSAMP,MAG,YARG,YSAMP,PI
      PI = 3.14159265
      TSAMP = 1.0
      I = 0
      J = 0
      L = 0
      WRITE(6,900)
900   FORMAT('  GENERATION OF DIRECT SEQUENCE BPSK SIGNAL',
     A          ' IN FILE bpsk.dat')
      WRITE(6,1000)
1000  FORMAT('  # SAMPLES TO GENERATE         = ',$)
c     READ(5, *)N
      print *,n
      WRITE(6,1001)
1001  FORMAT('  MAX AMPLITUDE OF SAMPLE VALUES (R)= ',$)
c     READ(5, *) MAG
      mag=amag
      print *,mag
      WRITE(6,1002)
1002  FORMAT('  # SAMPLES PER CARRIER CYCLE (R)= ',$)
c     READ(5, *) TCARR
      print *,tcarr
      WRITE(6,1003)
1003  FORMAT('  # SAMPLES PER CHIP BIT    (R) = ',$)
c     READ(5, *) TCHIP
      print *,tchip
      WRITE(6,1020)
```

```
1020 FORMAT('   ENTER (0):RANDOM CHIP, OR (1):REPEAT M-SEQ CHIP
=',$)
c    READ(5, *) ICHIP
     print *,ichip
     WRITE(6,1004)
1004 FORMAT('  # SAMPLES PER INFO BIT   (R) = ',$)
c    READ(5, *) TDATA
     print *,tdata
     WRITE(6,1005)
1005      FORMAT('  # DELAYS BEFORE SAMPLING   (R)= ',$)
c    READ(5, *)TDELAY
     print *,tdelay
     WRITE(6,1006)
1006 FORMAT('  RANDOM NUMBER SEED (I4)---> ',$)
     l=11
c    READ(5, *)L
     print *,l
c    Initialize RAND by calling with input non-zero L.
c    Subsequent calls will be with L = 0.
     RANDOM=RAND(L)
c    debug
C    WRITE(6,*) ' SEED AND RANDOM NUMBER RETURNED:',L,RANDOM
     L=0
     WRITE(6,1010)
1010 FORMAT(/1X,'------------------WORKING-------------------')
     IF (ICHIP .EQ. 0) GO TO 5
     OPEN (UNIT = 1, FILE = 'FSR.DAT', STATUS = 'OLD')
     READ (1,500) IFSR
     READ (1,600) (IFBD(K) , K = 1, IFSR)
500  FORMAT(I3)
600  FORMAT(I3)
     CLOSE (UNIT = 1)
     DO  5 K = 1 , IFSR
     IVAL(K) = 1
5    CONTINUE
     DO 100 K = 1,N
C  CHECK TO SEE IF WE NEED TO GENERATE A NEW DATA BIT
     IF ((K+TDELAY)*TSAMP .LT. I*TDATA*TSAMP) GO TO 10
C  IF SO DO IT HERE
     RANDOM = RAND(L)
c    debug
C    WRITE(6,*) ' IFLAG AND RANDOM NUMBER RETURNED:',L,RANDOM
     INFO = 0
     IF (RANDOM .GT. 0.5) INFO = 1
C  KEEP TRACK OF WHICH DATA BIT WE ARE ON
     I = I+1
10   CONTINUE
C  NOW CHECK TO SEE IF WE NEED TO GENERATE A NEW CHIP BIT
     IF ((K+TDELAY)*TSAMP .LT. J*TCHIP*TSAMP) GO TO 20
     IF (ICHIP .NE. 1) GO TO 15
     CHIP = IVAL(IFSR)
     CALL MSEQ(IFSR,IFBD,IVAL)
     GO TO 19
15   CONTINUE
C  IF SO DO IT HERE
     RANDOM = RAND(L)
     CHIP = 0
     IF (RANDOM .GT. 0.5) CHIP = 1
C  KEEP TRACK OF THE CHIP BIT NUMBER
19   CONTINUE
     J = J+1
20   CONTINUE
C  NOW WE DETERMINE THE PHASE SHIFT
     PHASE = FLOAT(MOD(INFO + CHIP,2))
C COMPUTE THE ARGUMENT FOR THE COSINE FUNCTION
     YARG = 2.0*PI*(1.0/(TCARR*TSAMP))*(K+TDELAY)  + PI*PHASE
     YSAMP(K)= MAG * COS(YARG)
100  CONTINUE
C NOW SAVE THE VALUE
c  ranga  OPEN (UNIT=1,FILE='bpsk.dat',STATUS='NEW')
c    CALL DATE(BDATE)
c    CALL TIME(CTIME)
c-ranga   WRITE(1,125)N,MAG,TCARR,TCHIP,TDATA
125     FORMAT(I5,5X,F4.1,10H*DSBPSK+M2,F7.2,1Hf,F7.2,1Hc,F7.1,
     *   1Hd)
c-ranga   WRITE(1,150) (YSAMP(I),I=1,N)

     do 121 i=1,n
c    write(1,*) ysamp(i)
121  continue

150  FORMAT(8F16.8)
     CLOSE (UNIT=1)
     return
     END

     SUBROUTINE MSEQ(IFSR,IFDB,IVAL)
C    THIS SUBROUTINE PERFORMS THE SHIFTING OPERATION OF AN IFSR
STAGE
C    FEEDBACK SHIFT REGISTER, WITH FEED BACK CONNECTIONS AS
INDICATED
C    BY IFDB.  IVAL IS THE INITIAL CONTENTS OF THE FSR AND WILL
CONTAIN
C    THE FINAL CONTENTS AFTER SHIFTING.
C              W.R. TUCKER 4 AUG 83
     INTEGER IFDB(IFSR), IVAL(IFSR)
     ISUM = 0
     DO 10 I = 1, IFSR
     ISUM = IFDB(I) * IVAL(I) + ISUM
10   CONTINUE
     IBIT = MOD(ISUM,2)
     DO 20 I = 1, IFSR - 1
     IVAL(IFSR + 1 - I) = IVAL(IFSR  - I )
20   CONTINUE
```

```
      IVAL(1) = IBIT
      RETURN
      END

c     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
c     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
c     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
c     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

REFERENCES

1. M.H.A.Davis*et al.,Stochaistic Modelling and Control*London, England: Chapman and Hall,1985.
2. P.Young. *Recursive Estimation and Time series Analysis* New York: Springer-Verlag,1984.,ch.6.,p. 118.
3. A.Albert,*Regression and the Moore-Penrose Pseudoinverse.* New York: Academic,1972 ,p.48.
4. D.M.Himmelblau *Applied Non-linear Programming.* New York: Mc- Graw-Hill, 1972,p.75.
5. G.Berchin, M.A.Soderstrand, and R.Roberts, *Deconvolution Using Model Based Parameter Estimation* Proc 1987 Midwest Symp on Circuits & Systems, Syracus, NY August 1987.
6. M.A.Soderstrand, K.V.Rangarao and G. Berchin, *Enhanced FDLS Algorithm for Parameter Based System Identification for Automatic Testing* Proc 1990 Midwest Symp on Circuits & Systems Calgary, Alberta Aug 1990.
7. D.Curtis Schleher *Introduction to Electronic Warfare.* Artech House Inc, 1990,p.326-330.
8. Tom Kwan and Kenneth Martin, *Adaptive Detection and Enhancement of Multiple Sinusoids Using a Cascade IIR Filter* IEEE Trans Circuits and Systems, vol 36 no 7 July 1989.
9. Mariane R. Petraglia, Sangit K Mitra, and J.Szczupak *Adaptive Sinusoid Detection Using IIR notch filters and multirate Techniques,* Proc. 1990 IEEE Intern. Symp. CAS, New Orleans, LA, pp 271-274, May 1990.
10. John J Shynk, Christina K Chan and Mariane R Petraglia, *Blind Adaptive Filtering in the Frequency Domain* , Proc. 1990 IEEE Intern. Symp. CAS, New Orleans, LA, pp 275-278, May 1990.
11. Mariane R petraglia, J.J.Shynk, and S.K.Mitra *Stability Bounds For an Adaptive IIR Notch Filter* Proc. 1990 IEEE Intern. Symp. CAS, New Orleans, LA, pp 1963-1966, May 1990.
12. R.W.Hamming, *Digital Filters* pp (4-5)., 1980 Prentice Hall Englewood Cliffs, New Jersey.
13. Alan V. Openheim, Alan S .Wilsky and IAN T. Young, *Signals and Systems* 1983 Prentice Hall, Englewood Cliffs, New Jersey.
14. Micahel A Soderstrand, H.H.Loomis, R.Gnanasekaran. *Pipelining Techniques for IIR Digital Filters* Proc IEEE Symp CAS New Orleans, LA pp(121-124) May 1990.

15. Torsten Söderström, Petre Stoica *System Identification* pp 97-112, 1989 Prentice Hall Englewood Cliffs, New Jersey.

16. Jerry M. Mendel *Lessons in Digital Estimation Theory* pp 88-91., 1987 Prentice Hall Englewood Cliffs, New Jersey.

17. P.Young. *Recursive Estimation and Time series Analysis* New York: Springer-Verlag,1984.,ch.6.,p. 108.

18. Robert D Strum., Donald E Kirk., *First Principles of Discrete Systems and Digital Signal Processing,* Addison-Wesley Publishing 1989.

19. Marple *Digital Spectral Analysis with Applications* Prentice-Hall Inc., Englewood Cliffs., New Jersey 1987.

20. Kay, *Modern Spectral Estimation Theory & applications* Prentice-Hall Inc., Englewood Cliffs., New Jersey 1988.

21. Simon Haykin, *Adaptive Filter Theory,* Prentice-Hall Inc., Englewood Cliffs., New Jersey 1991.

22. E. Oran Brigham *The Fast Fourier Transform and its applications* Prentice-Hall Inc., Englewood Cliffs., New Jersey 1988.

23. M.A.Soderstrand, T.G.Johnson and G.A.Clark, *Hardware Realizations of Frequency-Sampling Adaptive Filters* 1986 IEEE Intern. Symp. Circuits and Systems, San jose, CA, May 1986, pp 900-903.

24. M.A.Soderstrand and R.Miller *A moving recursive CNTT implemented in GQRNS arithmetic* 1989 IEEE Intern. Symp. on Circuits and Systems, Portland, OR, May 1989.

25. M.A.Soderstrand, H.H.Loomis and K.V.Rangarao, *Improved Real-Time Adaptive Detection, Enhancement, or Elimination of Multiple Sinusoids.* IEEE Midwest Symp. on Circuits and Systems, Monterey, CA May 14-17, 1991.

26. M.A.Soderstrand, H.H.Loomis and K.V.Rangarao. *Elimination of Narrow-Band Interference in BPSK- Modulated Signal Reception.* IEEE Intern. Symp. Circuits and Systems, Singapore, June 10-13.

27. G.Kane, *MIPS RISC Architecture,* Prentice Hall, 1989.

28. Honeywell manual on DSP chips HDSP66110 & HDSP66210.

29. M.A.Soderstrand, K.V.Rangarao, H.H.Loomis *New Algorithms for the Detection and Elimination of Sine waves and other Narrow-Band Signals in the presence of Broad-band Signals and Noise.* NPS technical report, September 1991.

30. John L Hennessy, David A Patterson *Computer Architecture A Quantitative Approach.*

31. , Morgan Kaufman Publication Inc, 1990, San Mateo, CA. WE DSP32C *Digital Signal Processor Information Manual,* AT&T Publication, 1990.

INITIAL DISTRIBUTION LIST

		No. of Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2.	Library, Code 52 Naval Postgraduate School Monterey, California 93943-5002	2
3.	Prof. Michael Soderstrand Department of EECS University of California Davis, California 95616	1
4.	Superintendent, Naval Postgraduate School Atten: Professor Herchael Loomis Jr, Code EC/Lm Naval Postgraduate School Monterey, California 93943-5000	1
5.	K.V.Rangarao, Dy.Director Naval Science & Technological Laboratory Visakapatnam, AP 530027, INDIA	2
6.	Director Naval Science & Technological Laboratory Visakapatuam, AP 530027, INDIA	1
7.	Director Directorate Training & Sponsored Research 233, B Block, Sena Bhavan New Delhi-110011, INDIA	3
8.	Director Defence Electronics Research Laboratory Chandrayana Gutta Lines Chandrayana Gutta Hyderabad, AP 500005, INDIA	1

9.	Dr. V.S.Arunachalam Scientific Advisor to Raksha Mantri and Secretary (R & D) DoD South Block, New Delhi 110011, INDIA	1